Modèles pour la création interactive intuitive

Adrien Bernhardt

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Informatique**

Arrêté ministérial :

Présentée par

Adrien Bernhardt

Thèse dirigée par **Marie-Paule Cani**

préparée au sein **LJK**
et de **MSTII**

Modèles pour la création interactive intuitive

Thèse soutenue publiquement le **mercredi 3 juillet 2013**,
devant le jury composé de :

Georges-Pierre Bonneau
Professeur, Université Joseph Fourrier, Grenoble, Président
Rafaëlle Chaine
Professeur, Université Claude Bernard Lyon I, Rapporteur
Laurent Grisoni
Professeur, Université Lille I, Rapporteur
Loic Barthe
Maitre de Conférence, Université Paul Sabatier, Toulouse, Examinateur
Gaël Guennebaud
Attaché de recherche, Inria Bordeaux, Examinateur
Marie-Paule Cani
Professeur, INPG, Grenoble, Directeur de thèse

Table des matières

Résumé

Cette thèse porte sur la création interactive intuitive de formes 3D, justifié pour les artistes par un besoin d'efficacité et d'expressivité et pour le grand public par la démocratisation d'usages de la modélisation numérique pour les jeux vidéo ou pour les imprimantes 3D. Dans ce cadre, pour trouver des nouvelles formes d'interactions, nous avons développé des modeleurs et des techniques de modélisation : un modeleur de formes libres 3D par métaphore de peinture, un modeleur vectoriel temps réel de paysages ainsi qu'un modeleur de paysages par croquis vu de la première personne. Les contributions scientifiques vont d'un opérateur de mélange implicite qui garanti la localité du mélange, à l'utilisation qu'une formulation bi-harmonique pour la modélisation de terrains permettant une modélisation fine, intuitive et temps réel de terrains ; en passant par la modélisation de paysages à partir de croquis composé de silhouettes vus de la première personne.

Abstract

This thesis focuses on the interactive intuitive creation of 3D Models. Improvment of modeling tools for artists is still an active research mater especialy concerning efficiency and expressiveness. Moreother video games or 3D printers are make the use of modeling tools more common by the general public. In this context, to find new forms of interaction, we have developed modelers and modeling techniques : a 3D free-form modeler which uses a painting metaphor for creation, a vector-based real-time landscapes modeler as well as a sketch-based landscapes modeler that uses first person drawnings. Scientific contributions range from a blending operator for implicit modeling that guarantees the locality of the blend. The use for a bi-harmonic formulation for landscape modeling that allow fine, intuitive and real-time terrain modeling. Finally through landscape modelisation using sketches composed of silhouettes seen from the first person point of view.

Introduction

Cette thèse s'intéresse à un sous domaine en particulier de l'informatique graphique : la modélisation géométrique. En particulier, la modélisation de formes arbitraires, mais aussi de terrains sont les deux principales applications que nous avons développé durant cette thèse. Mais l'aspect modélisation n'est pas l'unique point que nous avons travaillé. Nous avons porté une attention toute particulière à l'interface utilisateur, à l'expérience utilisateur que nos applications offraient, dans le but d'arriver à créer des modeleurs de qualité. Des modeleurs qui ne sont pas de simples interfaces d'algorithmes mais des environnements cohérents, multi-usages, propices à la création numérique.

Dans ce chapitre d'introduction nous souhaitons dans une première partie justifier rapidement notre choix de travailler l'expérience utilisateur puis dans une seconde partie présenter succinctement les principales contributions scientifiques ainsi que la structure du reste de ce document.

J'espère avec ce travail, réussir à vous faire partager le plaisir que j'ai eu à penser le processus créatif et à réaliser des prototypes à même de valoriser ce dernier.

Avancées des techniques et des usages en informatique graphique

L'informatique graphique permet de vivre des aventures extraordinaires, qu'il serait impossible de réaliser avec des effets spéciaux traditionnels. De Toys Stories à Avatar, les techniques qu'elle offre nous permettent de découvrir des

personnages, des mondes et des paysages tout droit sortis de l'imagination d'artistes ; des mondes chaque fois plus complexes, réalistes et originaux. Sans l'informatique graphique, ces réalisations seraient impossibles ; tout du moins pour un coût acceptable.

C'est avant tout la combinaison de trois domaines qui rend les effets spéciaux possible : les modèles géométriques, l'animation, et les techniques de rendu. A l'époque des débuts de l'informatique graphique, il n'était pas seulement question de faciliter la création graphique, il était question de la rendre possible, de réussir à produire sur ordinateur les idées sorties de la tête des artistes, difficiles à mettre en œuvre physiquement. Le domaine de l'informatique graphique s'est alors appuyé sur des modèles mathématiques pour la géométrie, pour le rendu, pour créer les premières applications. Les surfaces triangulaires et quadrangulaires, par exemple, commencèrent à être utilisées. Les surfaces implicites permirent de modéliser des objets et des effets volumiques alors impossibles à réaliser avec des triangles, les surfaces de Bézier permirent de modéliser des surfaces lisses. Ces modèles mathématiques étaient au début simples mais offraient de nouvelles possibilités. Les techniques ont été depuis, progressivement améliorées pour ouvrir le champ des possibilités et aussi simplifier la création. Les surfaces de subdivision ont été introduite. La théorie a aussi été améliorée pour mieux modéliser le transport de la lumière ou bien les propriétés des objets géométriques, grâce à la géométrie discrète différentielle. Cette dernière apporte un modèle unifié pour modéliser les surface discrète lisses, permettant, par exemple, tout une gamme d'opérations avancées sur les maillages triangulaires. Au lieu de contrôler, un à un, les triangles d'un maillage les artistes peuvent maintenant contrôler des surfaces qui se subdivisent automatiquement, permettant de manipuler ces surfaces lisses avec un nombre réduit de points de contrôle. En résumé, le domaine a énormément progressé en 30 ans.

Néanmoins les outils ainsi créés restent bien souvent très liés aux "<mathématiques"> qui régissent leur fonctionnement. Prenons l'exemple de la modélisation d'un objet à l'aide de surfaces de subdivision. Jusqu'à récemment il était encore difficile de contrôler certaines configurations de subdivision impliquant des arrêtes vives. Ce type de limitation a un impact direct sur l'utilisateur, qui doit s'adapter aux contraintes du modèle, en attendant que l'outil mathématique progresse. Les logiciels de modélisation enrobent les modèles mathématiques pour offrir une meilleur interface à l'utilisateur. Ce faisant ils ne peuvent palier tous les défauts des modèles qu'ils utilisent. Pour

cette raison, la modélisation 3D requiert de la part des artistes, une bonne connaissance des outils, du travail et du temps pour les maitriser.

Il n'est néanmoins pas choquant, pour un artiste, de devoir travailler pour maitriser une technique, un outil ou un logiciel. Le dessin, la peinture et la sculpture, ainsi que tous les autres domaines de la création, impliquent d'apprendre telle ou telle technique pour réussir à réaliser un effet particulier. Pourquoi alors, est-il dommageable d'offrir aux artistes des outils qui nécessitent du temps pour être maitrisés ? La raison est que le temps et l'effort que l'utilisateur consacre à surmonter les limitations d'un modèle est autant de temps qui est perdu pour la création et susceptible de rebuter les utilisateurs les moins persévérants.

Au final on peut considérer qu'il est maintenant possible de tout faire en informatique graphique, mais de quelle manière ? On peut encore trouver des modeleurs et des modèles plus simples et plus intuitifs à utiliser. C'est cette voie que nous avons choisi d'explorer dans cette thèse.

Objectif de cette thèse

Ce que j'ai souhaité explorer dans ma thèse est la modélisation instinctive ou : comment fournir à l'utilisateur un modeleur qui lui permette d'exercer directement sa créativité, sans croquis préalable, avec le minimum de planification. Bien que le croquis fasse partie du titre original de cette thèse, il n'est pas nécessaire d'avoir une interface par croquis pour obtenir une bonne expérience utilisateur. C'est par-contre l'essence du croquis que nous souhaitons apporter à l'utilisateur en terme d'interaction. Dans son livre "Sketching user Experiences" [30] Bill Buxton explique que le croquis est un langage, ce qui semble logique. Mais en plus de cela l'action de croquer est comme un dialogue entre le dessinateur et son dessin. Un dessin incomplet est suggestif, les éléments manquants sont évocateurs et sont créés virtuellement dans la tête du créateur. C'est ainsi que le dessin en cours de construction donne en quelque sorte des idées au dessinateur. Ensuite le dessinateur met ses idées sur papier. Ainsi une conversation s'établit entre l'utilisateur et son dessin. C'est entre autres ce processus que nous avons souhaité explorer durant cette thèse. Notre processus créatif passe donc par une boucle qui cherche à combiner action et interprétation. Une action peut en engendrer une nouvelle,

mais elle peut aussi engendrer une correction.

L'avantage d'un croquis est qu'il est évocateur, ce qui est difficile à retranscrire dans un modeleur. Que ce passe t'il alors quand l'utilisateur s'attend à avoir un résultat précis, celui que lui évoque son action, et qu'il est déçu par le résultat ? Si l'utilisateur doit recommencer son action depuis le début sans autre validation que le résultat final alors il va rapidement se frustrer. Pour palier ce problème deux solutions sont possibles : la correction et l'accompagnement. Faciliter la correction d'une action utilisateur permet de lui éviter de refaire son action en entier. Au lieu de cela il peut par exemple corriger localement son dessin, uniquement là où il le juge nécessaire. L'accompagnement consiste à offrir un retour perceptif en temps réel de l'action en cours d'exécution de façon à permettre un ajustement du geste au cours de l'action. Cette technique nécessite un retour visuel en temps réel. Imaginez un simulateur chirurgical qui ne fournisse un retour visuel et haptique que toutes les secondes. A l'opposé si l'utilisateur souhaite avoir un contrôle grossier sur une action alors un retour en temps interactif (toutes les secondes) est acceptable.

Une constante de ce travail de thèse est que j'ai toujours souhaité travailler dans le sens de la création numérique en me basant sur les qualités du croquis pour construire des modeleurs interactifs, avec le sentiment que la qualité de l'expérience utilisateur prime sur le nombre des fonctionnalités. À l'origine le sujet de cette thèse était le suivant : "Création de mondes virtuels : Combiner croquis et Modèle procéduraux". Ce sujet a légèrement changé au cours de cette thèse de manière à troquer le croquis pour l'ébauche ; qui a une acception plus large. C'est à dire que le croquis n'est pas uniquement le fait de dessiner c'est un processus de création qui permet de progressivement raffiner une idée, de construire progressivement sur papier, ou sur tout autre support, un raisonnement de pensée.

Vue d'ensemble du reste du document

Dans le chapitre suivant nous présentons l'État de l'Art des techniques existantes concernant la modélisation et la création intuitive.

Le reste du document est séparé en deux grandes parties. La première traite

de la modélisation volumique de formes arbitraires à l'aide de surfaces implicites. Nous y présentons MATISS, un système de modélisation par peinture ainsi qu'une technique permettant de rendre la modélisation implicite plus intuitive en limitant le mélange de surfaces implicites aux zones qui s'intersectent.

La seconde partie est consacrée à la modélisation surfacique de terrains. Dans cette partie nous utilisons des "<équations différentielles"> et un solveur sur GPU pour créer des systèmes de modélisation temps réel. Le premier chapitre de cette seconde partie explique comment fonctionne le modeleur de terrain et ce qu'il est possible de faire avec. Le second chapitre propose une interface de modélisation de paysages "<à la première personne"> nommée "<dessiner des montagnes">.

Pour finir la conclusion contient une synthèse du travail effectué dans cette thèse et expose quelques perspectives du domaine de la création numérique.

Contributions

Dans cette thèse nous avons plusieurs contributions et nous avons participé à quelques travaux de recherche.

Le travail MATISS[4] présenté dans le chapitre 2 a été publié dans la conférence SBIM en 2008. Le travail sur le mélange local de surfaces implicites [1] a été présenté à Eurographics en 2010 et découle d'une problématique soulevée dans MATISS. Ce travail à lui même été l'objet d'une extension par Gourmel[6] publiée dans TOG en 2013 à laquelle j'ai participé.

Le travail sur la modélisation de terrain a fait l'objet d'un poster SIGGRAPH [3] en 2011 ainsi que d'une publication à SIBGRAPI la même année [2]. Les travaux sur la formulation bi-harmonique de l'édition de terrain, l'édition sur terrain existant et la modélisation de terrain par croquis n'ont pour l'instant par encore été publiés. Pour finir j'ai contribué à une publication de Emilien [5] sur la génération procédurale de villages.

Chapitre 1

État de l'art

L'état de l'art est une partie incontournable d'un manuscrit de thèse, et vise à présenter l'état actuel des connaissances en relation avec le sujet de la thèse. La finalité principale est de donner au lecteur les outils pour comprendre le travail présenté dans le reste du document. Nous présenterons donc les domaines de la modélisation implicite et de la modélisation de terrains, puisqu'ils sont les domaines principaux abordés dans cette thèse, ainsi que des techniques utilisées pour rendre une expérience de modélisation plus intuitive.

Cette dernière partie mérite peut-être une petite explication. A l'origine cette thèse devait porter précisément sur la modélisation par croquis. Le croquis contribue à rendre la modélisation intuitive mais n'englobe pas tous les aspects de cette dernière. C'est pourquoi la partie sur la modélisation intuitive contient majoritairement des références à la modélisation par croquis, mais aussi des références à d'autres travaux qui nous ont inspiré ou interpelé.

La première partie, "<Techniques de modélisation"> (Section 1.1), contient un état de l'art axé sur les outils que nous avons utilisés dans cette thèse. Une première sous partie traite des surfaces implicites. Une seconde traite de la modélisation de terrain, tandis qu'une dernière regroupe quelques travaux notables, qui nous ont inspiré et ne pouvant pas figurer dans ces deux catégories.

La seconde partie , "<modélisation intuitive"> (Section 1.2), aborde les techniques numériques de création intuitive et de modélisation. Cette partie est

découpée en trois sous-parties. Une première sous-partie analyse quelques exemples de modélisation intuitive issus du monde réel. La seconde partie est dédiée aux interfaces par croquis tandis que la dernière regroupe des travaux ne possédant pas vraiment d'interface par croquis, mais qui nous ont inspiré.

1.1 Techniques de Modélisation

Nous présentons ici des techniques de modélisation liés aux travaux effectués durant cette thèse. Nous présentons d'abord des travaux sur la modélisation implicite, puis sur la modélisation de terrain. Enfin nous présentons quelques travaux qui nous ont particulièrement inspiré.

1.1.1 Modélisation par Surfaces Implicites

Les surfaces implicites sont une représentation particulière des surfaces. Son utilisation n'est pas très répandue en modélisation générale de surface, mais est plus courante et efficace pour certaines applications. En particulier les applications nécessitant une représentation volumique. On peut citer le rendu de simulation de fluides [40, 85] ou l'extraction de surface distance autour d'une forme [64].

Une surface implicite est définie comme une iso-surface d'un champ potentiel. C'est à dire que ce sont l'ensemble des points vérifiant une équation du type $f(P) = C$ ou f est une fonction de champ ayant un support local ou global. D'autres terminologies ont été utilisées pour ces fonctions telles que champ potentiel dans le cas de l'utilisation de squelette pour la modélisation, ou encore fonctions implicites. Cette dernière appellation, bien que couramment utilisée est inappropriée car les fonctions sont tout à fait explicites, seules les surfaces sont implicites. Nous préférons le terme fonction de champ qui exprime clairement que la surface implicite est une iso-surface du champ scalaire qui la définit.

La fonction de champ peut être définie de différentes manières : par une level-set, c'est à dire un front de propagation dans une grille [68, 57], ou par une représentation fonctionnelle comme avec les f-reps[58], soft objects [83],

les fonctions distance, les surfaces de convolution [69], les RBF [63] et MLS [49, 7].

Classification des types de fonctions de champ

Il est important de comprendre que chacune de ces représentations possède des propriétés spécifiques qui obligent de choisir des opérateurs de composition différent pour garder une fonction de champs qui convienne pour des mélanges à venir.

Support Local/ Support Global : On dit que le support d'une fonction de champ est local quand le champ s'annule au delà d'une certaine distance. A l'opposé une fonction à support global ne s'annule pas partout, là aussi au delà d'une certaine distance. La plupart des fonctions de champs à support global se comportent comme des distances à la surface implicite : elles sont négatives à l'intérieur, égales à zéro au niveau de la surface et tendent vers l'infini quand on s'éloigne, c'est le cas des f-rep[58]. On appelle aussi fonction à support global les fonctions de champ positives qui tendent vers zéro avec la distance mais ne s'annulent jamais. Certaines surfaces de convolution, que nous présentons un peu plus loin, utilisant un noyau de convolution à support global, sont dans ce cas.

Modélisation par squelette : Beaucoup de techniques de modélisation implicite reposent sur un champ généré par des primitives squelettes (points, segments, triangles, etc). La fonction de champ est alors aussi appelée champ potentiel. Dans ce cas la fonction de champ est généralement une fonction positive qui décroit vers zéro lorsque l'on s'éloigne des primitives squelettes. L'ensemble des primitives squelettes est aussi appelé tout simplement squelette de la surface implicite. L'iso-valeur C est alors strictement positive, et la fonction de champ est plus importante à l'intérieur du volume qu'à l'extérieur. Les surfaces implicites à support local telles que les meta-ball[20] et les soft-objets[83] appartiennent à cette catégorie ; leur potentiel s'annule au-delà d'une distance finie au squelette.

La modélisation par squelette est l'un des grands avantages des surfaces implicites, car intuitive pour l'utilisateur et facile à mettre en œuvre de manière

naïve : Il suffit de combiner les fonctions de champ de chaque primitive pour obtenir la fonction de champ résultante.

Surfaces de convolution : Bien qu'il soit facile de combiner naïvement des fonctions de champ, l'opération peut se révéler délicate car des gonflements peuvent apparaitre lorsque l'on mélange des primitives squelettes qui sont trop proches. Si le mélange se fait de manière additive, on a intérêt à ce que les potentiels soient eux même définis de manière additive. Les surfaces dites "<de convolution">, introduites par Bloomenthal [22], qui consistent à convoluer un noyau de convolution avec un squelette, dit de convolution, résolvent en partie ce problème. L'intégrale étant une opération additive sur le domaine d'intégration (ici le squelette), l'intégrale de convolution calculée sur le squelette entier est donc égale à la somme des intégrales de convolution des éléments du squelette. Grâce à cette propriété deux segments de convolution se touchant, ne gonflent plus au niveau de leur jonction.

Applications à la modélisation

Les surfaces implicites ont été utilisées dans plusieurs travaux de modélisation de surfaces. On peut noter en particulier les travaux reposant sur le blob-tree [80, 67, 66], les travaux de sculpture virtuelle [35][60], ainsi que les travaux permettant l'édition interactive de squelettes implicites [12].

Limitations

Mélange à distance : Un problème bien connu de la modélisation implicite est le mélange non-désirable. Quand l'utilisateur crée un modèle complexe en mélangeant successivement des primitives, ces dernières risquent souvent de se mélanger alors que l'utilisateur ne le souhaite pas. Par exemple modéliser le bras d'un personnage qui se mélange au niveau de l'épaule mais reste proche du corps sans se mélanger partout ailleurs est difficile. Faire cela nécessite souvent l'utilisation de mécanismes complexes tels que des graphes de mélange ou bien des fonctions de décroissance [13]. Heureusement, l'opérateur de mélange local que nous utilisons dans notre application nous permet de mettre en place une solution intuitive plus simple.

Absorption des petits détails par les grands : Le problème connu, lors d'un mélange, de l'absorption des petits détails par les grands [82], arrive quand l'objet comportant les petits détails, gonfle lors du mélange. C'est ce gonflement qui conduit à l'effacement des petits détails. L'opérateur arc-d'une-ellipse, défini dans [16], permet de résoudre ce problème en contrôlant le mélange indépendamment de la taille de chaque objet. Cet opérateur, aussi appelé opérateur de Barthe, tire son nom du fait que le mélange prend la forme d'un arc d'ellipse lorsque combine des champs potentiels de gradients constants. Le second opérateur présenté dans [16] offre pour chacun des objets mélangés, un paramètre qui permet de limiter l'extension du mélange, ce qui permet de limiter l'effacement des petits détails.

1.1.2 Techniques de modélisation de terrains

Dans la première partie nous avons abordé la modélisation volumique à l'aide de surfaces implicites. Nous abordons maintenant les techniques de modélisation de surfaces, en particulier de terrains car c'est le sujet central de la seconde partie de cette thèse. C'est un domaine composé de techniques qui peuvent varier beaucoup d'un travail de recherche à un autre. Presque tous les travaux de modélisation de terrains réduisent la dimensionnalité du problème en considérant qu'un terrain ne comporte généralement pas de surplomb ; cela permet de modéliser le terrain en tant que carte d'élévation ou DEM (Digital Elevation Model). C'est pourquoi l'éditeur de terrains de FarCry propose de transformer par endroits le terrain surfacique en terrain volumique pour avoir le meilleur des deux mondes. Peu de travaux utilisent des représentations volumique car elle sont généralement trop couteuses. Une exception notable est le système Arches[59] qui propose un modèle permettant astucieusement de définir plusieurs couches de matériaux, et ainsi créer des surplombs à l'aide de couches d'air, sans le surcout d'une grille 3D pleine. Il s'agit en fait d'une grille 3D avec compression [18] sur z.

Fractales : Les terrains, ont une nature fractale qui est exploitée dans plusieurs travaux de modélisation de terrain ; à commencer par le papier fondateur de [53], et les logiciels d'édition de terrain [34],[71]. L'avantage des fractals est qu'il est possible, avec peu de paramètres de générer des paysages réalistes. L'inconvénient est qu'il est souvent difficile de fixer ces paramètres

pour obtenir un effet désiré. L'utilisation d'une subdivision contrainte [17] donne une piste pour surmonter cette difficulté mais étendre cette approche à d'autres types de terrains reste difficile. Enfin le travail [32] utilise des bruits procéduraux pour permettre à l'utilisateur de peindre interactivement un terrain à l'aide de pinceaux.

Exemples Réels : Les deux approches citées utilisent de la génération de texture pour arriver au résultat. La première méthode [27] permet de générer une texture de terrain (carte d'élévation) à partir d'exemples, mais aucun contrôle n'est disponible. L'algorithme présenté dans [86] permet par génération de texture (de carte d'élévation) de créer un terrain similaire à un terrain de référence. La méthode présentée permet de dessiner sur une carte la position des principales chaines de montagnes ou canions que l'on souhaite créer. Le terrain résultat est alors assemblé à partir de petits morceaux du terrain donné en exemple. Cette méthode donne des résultats très convainquant mais nécessite un grand temps de calcul et n'offre pas un contrôle fin de la forme de la montagne résultante.

Édition Vectorielle : D'autres travaux permettent l'édition vectorielle de terrains. Le logiciel Proland [29][28] permet l'édition en temps réel de données vectorielles à la surface du terrain, mais ne permet pas d'éditer le terrain en tant que tel. Par-contre [42] permet, grâce à la résolution d'une équation de diffusion, la génération de terrains à partir de contraintes vectorielles. La méthode présentée est rapide et offre divers contrôles sur l'altitude du terrain en un point comme sur le gradient de la pente à proximité des contraintes.

Simulation Physique : Il est aussi possible de simuler les processus naturels en jeux, érosion, sédimentation, pour générer un terrain réaliste[53]. Ce genre de traitement, à la fois très couteux et très parallélisable a beaucoup bénéficié de la puissance de calcul des GPUs [76][14].

Interface par Croquis : Pour finir trois travaux notables en modélisation par croquis. Un mécanisme très simple de déformation de terrain était présenté dans le travail précurseur Harold[31]. Les papiers [77][37] présentent des mécanismes d'édition de DEM à partir de croquis. [37] permet entre autre

d'extraire les paramètres de bruit procédural du trait dessiné par l'utilisateur. La section 1.2.2 est toutefois dédiée à la modélisation par croquis.

1.1.3 Autres travaux

Certains travaux, qui ne sont pas directement liés à la modélisation de terrain, nous ont beaucoup inspiré ; nous les présentons ici.

Géométrie discrète différentielle : cette théorie permet d'appliquer la théorie de la géométrie différentielle aux surfaces triangulaires ou quadrangulaires [51]. Les opérateurs différentiels tels que le Laplacien permettent de caractériser les propriétés d'une surface telle que sa courbure en un point, relativement à ces voisins. Cette formalisation permet une large game d'opérations comme la déformation [72][25], le transfert de détails d'une surface sur une autre [72], la compression de maillage, l'édition vectorielle de maillages [54][87].

Diffusion curves : Ce travail, [56], porte sur le dessin vectoriel et le révolutionne. Il est particulièrement intéressant car des opérateurs différentiels sont aussi à l'œuvre ici. Le dessin est généré par diffusion de la couleur autour des courbes vectorielles. L'utilisation de l'équation de diffusion permet d'interpoler les couleurs sans avoir besoin de maillage ; la résolution multigrille sur GPU de l'équation différentielle est aussi particulièrement efficace. Pour finir le travail de [19] montre qu'il est possible, à partir de modifications simples, d'enrichir significativement en fonctionnalités les diffusion curves.

Modélisation par blocs : Enfin le papier de Leblanc[48] est intéressant car il combine modélisation volumique, surfacique tout en gérant correctement les coordonnées de textures.

1.2 Modélisation Intuitive

1.2.1 Dans la vie réelle

L'homme crée depuis "<l'aube des ages">, et cherche depuis longtemps des outils pour exercer ce besoin de création. Ne peut-on donc pas aller chercher de l'inspiration dans ce qui existe depuis longtemps, bien avant que l'informatique ne soit inventée ? C'est ce que nous allons faire ici en développant rapidement trois points : Le papier/crayon, la sculpture et le design.

Papier/Crayon : C'est certainement l'outil de création le plus emblématique et universel. Cet outil a cela de formidable, car sous une forme ou une autre, il existe depuis toujours et est toujours d'actualité : Un bâton avec du sable, du charbon sur un mur et aujourd'hui une tablette tactile ou graphique.

Le croquis est un moyen de création privilégié car il permet une sorte de dialogue entre le dessinateur et son dessin. Ce dialogue est particulièrement riche ; l'artiste a beaucoup de mots à son vocabulaire, et il est libre de la syntaxe. Il peut choisir de dessiner un petit détail avant les gros. Il peut alterner aisément entre plusieurs échelles de traits, de détails, sans que cela ne lui impose de contraintes. Le dessin est tolérant, il supporte les erreurs et les imprécisions, en restent compréhensible. Le dessin peut-être effacé, réinterprété.

Un des grands avantage du dessin c'est qu'il permet de structurer son raisonnement ; il permet de raffiner une idée, de commencer par établir les grandes lignes d'une œuvre. L'artiste peut sans contraintes travailler une à une, séparément, les sous parties de son dessin.

Le dessin au crayon est accessible à tous, les enfants prennent du plaisir à dessiner et arrivent à des résultats qui les satisfont, malgré leur manque de technique. A l'opposé, les personnes coutumières du dessin peuvent développer à l'aide du même outil des techniques avancées, des processus de dessin élaborés. Peu d'outils informatiques peuvent se vanter de s'adapter aussi bien aux experts et aux novices.

Argile/Sculpture : Vient ensuite l'argile ou la sculpture. Tandis que le dessin semble propice à l'improvisation, cela est déjà moins le cas pour la sculpture ou la poterie. Le matériaux impose plus de contraintes au créateur, et la planification est donc plus importante. Sur un bloc de pierre qui a été trop creusé il n'est pas possible de revenir en arrière. De même pour la poterie il est important de ne pas se tromper dans la forme globale au début car elle sera d'autant plus difficile à ajuster que la quantité de matière à déplacer est importante. En un mot la sculpture requiers plus de méthode ; l'improvisation est plus contraignante qu'avec le dessin.

La sculpture permet aussi de créer des formes solides, tridimensionnelles, réelles, qui ne sont pas uniquement des représentations mentales. D'un autre coté c'est une discipline qui favorise moins la créativité instinctive car il impose une plus grande planification, due aux contraintes imposées par les matériaux.

Design : La créativité n'est pas seulement nécessaire dans le domaine des arts ; elle est aussi essentielle en informatique, mathématiques, architecture pour ne citer que quelques exemples. Mais s'il y a un domaine qui s'intéresse en particulier à la création c'est le design ; le design regorge justement de techniques pour favoriser la création, la maturation et l'ébauche d'idées. Le livre "<Sketching User Experiences"> [30] de Bill Buxton aborde deux problèmes intimements liés à la création et au design. Le premier problème est de savoir qu'est-ce qu'une bonne interface utilisateur. Et pour cela il n'utilise pas la notion d'interface mais plutôt celle d' "<Expérience Utilisateur">. L'expérience utilisateur est une notion qui peut paraitre floue mais qui tend à prendre en compte l'expérience vécue par l'utilisateur lors de l'utilisation d'un produit, afin d'aider les concepteurs de ce produit à l'améliorer. De quels types de produits parlons-nous ? Des produits pouvant aller de la publicité, au presse agrume, en passant par des voitures, bâtiments, de l'électronique grand public, des logiciels. C'est ici que ce livre est intéressant : il s'adresse à un large éventail de créateurs, pour ne pas dire à tout le monde.

La première partie "<User Experiences"> du titre exprime que pour bien concevoir un produit il faut se concentrer sur l'expérience utilisateur ; ce qui est pour moi très important au sein d'outils de création comme ceux que nous souhaitons développer.

Pour créer de telles expériences utilisateurs, le livre présente entre autre des méthodes utilisées par les designers dans leur travail. Le point commun entre les méthodes présentées est qu'elles permettent l'ébauche de design. L'ébauche c'est justement le premier mot du titre du livre : "<Sketching">, pris dans son acception large. On traduit souvent le mot "<Sketching"> par "<Croquis"> ; c'est vrai mais cela a un sens beaucoup plus pauvre que celui du mot "<Ébauche"> qui peut s'appliquer à des dessins, des objets, des processus, des programmes. Le livre présente donc les qualités qui font du croquis une technique parfaite pour l'ébauche ainsi que d'autres techniques de création possédant aussi ces qualités.

Ces aspects du livre de Buxton ont contribué à forger notre idée de ce qu'est un système de modélisation adapté à la création.

1.2.2 Interfaces par Croquis

Intéressons nous maintenant à la création intuitive en informatique, en particulier aux interfaces par croquis. Ces dernières sont populaires en informatique graphique car elles permettent de créer des outils plus intuitifs et rapides à utiliser. Elles sont la promesse de pouvoir pourvoir l'informatique de certaines qualités qui lui font souvent défaut : facilité et rapidité de prise en main, amélioration de la créativité et de l'expressivité.

Les interfaces par croquis ont été utilisées dans beaucoup de domaines de l'informatique graphique : pour la déformation, l'animation, la pose de personnages, pour les vêtements, les paysages, la végétation, l'esquisse de formes 3D, etc. Certains travaux s'en servent comme une interface pour commander des actions avec toutefois une expressivité accrue. C'est le cas pour plusieurs travaux de déformation de maillage[55] ou pour le positionnement/manipulation des objets en 3D [65].

En ce qui concerne notre travail, on peut classifier les types d'interfaces par croquis en 2 grandes catégories. La modélisation de formes libres d'une part, pour lesquelles on ne connait pas l'objet modélisé. D'autre part les systèmes de modélisation par croquis utilisant des connaissances a priori, ce qui permet d'extraire plus d'informations des traits dessinés par l'utilisateur. Nous présentons ces deux catégories ainsi qu'un paragraphe sur les systèmes capables de traiter des croquis composés de plusieurs traits et un petit paragraphe sur

l'interprétation de croquis.

Modélisation de formes libres : Ces travaux reposent souvent sur le même principe : l'homme, lorsqu'il aperçoit la silhouette d'une forme qu'il ne connait pas, imagine la forme la plus simple possible possédant cette silhouette. Cette forme est ronde et son épaisseur est proportionnelle à son diamètre. Le papier fondateur dans ce domaine, Teddy[44], proposait une interface à base de gestes pour créer, découper, déformer, ajouter des parties. La géométrie est obtenue en manipulant directement le maillage. Le papier ShapeShop[67] qui présente un système qui utilise la modélisation variationnelle implicite comme représentation, propose une interface qui permet de manipuler interactivement le modèle crée. Beaucoup d'outils sont offerts à l'utilisateur qui bénéficie d'une interface par croquis pour manipuler sa création ainsi. Le travail de Rivers[62] propose de générer les formes à partir des silhouettes dans les directions X, Y et Z. La surface est générée grâce à la géométrie discrète différentielle à partir de ces trois silhouettes.

Bien qu'astucieuses ces approchent ne permettent pas de créer une variété d'objets suffisante pour pouvoir se passer d'une étape d'assemblage ou d'édition des éléments créés par croquis.

Utilisation de connaissances a priori : Une seconde catégorie de modeleurs par croquis existe aussi. C'est la catégorie des modeleurs spécialisés. Ces derniers intègrent des connaissances a priori fortes sur l'objet modélisé afin de pouvoir extraire plus d'information du croquis saisi par l'utilisateur. De telles approches ont été utilisées pour une vaste catégorie de modeleur et offrent une grande qualité de résultat, au détriment de la généralité. Parmis les domaines d'application on peut citer les plantes [11], les arbres [79], les vêtements [75], terrain[37], les cheveux [78] ou encore une catégorie plus large de formes à l'aide de patrons[84]·

Croquis multi-traits : L'analyse automatique des relations entre les différents traits d'un dessin est un sujet difficile car il faut que le système soit capable de comprendre les relations entre les traits. Cela est cependant intéressant car cela peut éviter à l'utilisateur de devoir assembler et positionner chaque partie qu'il dessine. Peu de système permettent de traiter ce genre de

croquis. Karpenko [46], propose un logiciel de modélisation libre de surface capable de construire un modèle à partir d'un croquis composé de plusieurs traits ; les traitement des relation entre traits est élégant mais ad-hoc. Dans le cadre de la déformation à partir de croquis de silhouette, Kraevoy[47] propose une approche variationnelle capable de traiter plusieurs traits en même temps. Gingold [38] propose lui de construire la forme 3D en laissant l'utilisateur annoter les relations entre les composantes du dessin, sans jamais changer de point de vue.

Interprétation de croquis : L'interprétation de croquis est aussi un domaine de recherche actif utile par exemple pour la reconnaissance d'écriture, de formules mathématiques, de schéma électriques. Parmi ces travaux deux m'ont intéressé en particulier. Le premier de Alvarado[10] propose d'interpréter des croquis à l'aide de réseaux Bayesiens. Ce qui est particulièrement élégant est qu'il est possible avec cette méthode de définir une grammaire de forme, ce qui permet de proposer une approche robuste est fonctionnelle de l'interprétation de croquis. Une autre approche [45], utilise un système multi-agent pour l'interprétation de croquis architecturaux. L'avantage de ces méthodes est qu'elles laissent à l'utilisateur une grande liberté d'action, ils peuvent dessiner des parties qui, si elle ne sont pas comprises par le système, seront tout simplement ignorées.

Première partie

Modélisation Volumique

Chapitre 2

Design par croquis de formes arbitraires

2.1 Introduction

Dans ce chapitre nous traitons de la modélisation de formes arbitraires, c'est à dire de formes n'ayant pas de signification sémantique exploitable. Ces formes répondent néanmoins à certaines règles : ce sont des volumes fermés (par opposition à des vêtements) et lisses (sans arrêtes vives). Ce travail, qui propose une interface par croquis, se base sur des travaux réalisés en perception qui montrent que l'être humain, lorsqu'il regarde une silhouette dont il ne reconnait pas la forme, imagine cette forme, comme étant la plus simple possible. Si la silhouette ne comporte pas d'arrêtes vives, l'humain imagine alors une forme lisse, la plus symétrique possible, passant par cette silhouette. Cette forme coïncide alors avec l'union des sphères maximales formant cette silhouette. Cette approche n'est pas vraiment nouvelle puisque qu'elle fut la base de Teddy [44], travail précurseur sur la modélisation de formes arbitraires par croquis, qui fut suivi par nombre d'autres travaux. Ces derniers ont en commun la volonté de démocratiser la modélisation de formes libres, de la rendre accessible à quiconque. Le système Teddy a montré à la communauté qu'une interface astucieuse, à base de croquis, peut formidablement simplifier non seulement la création initiale de formes 3D mais aussi l'extrusion, la découpe, la déformation ; rendant ainsi la création 3D accessible aux

novices.

Ce chapitre présente donc un travail de début de thèse sur la modélisation libre de formes 3D réalisé en collaboration avec Adeline Pihuit. Adeline a particulièrement travaillé sur la génération de bouts de formes élémentaires à l'aide de surfaces implicites de convolution, tandis que je me suis concentré sur l'interface du modeleur et le mélange successif de ces éléments. Ce travail a donné lieu à publication dans la conférence SBIM en 2008 et le logiciel a été utilisé à plusieurs reprises pour des démonstrations : lors de salons et d'expositions comme "<Remue Méninges">, destinés aux élèves de primaire ou lors de la fête de la science.

Comparé aux travaux précédents, ce travail comporte certaines caractéristiques d'interface et de modélisation qui permettent d'offrir une expérience utilisateur originale. Nous utilisons par exemple une interface par peinture plutôt que par croquis ; cela simplifie la création de composants de genre topologique arbitraire. Nous proposons un mélange lisse mais localisé aux zones d'intersection entre composants, grâce à une solution hybride exploitant une double représentation implicite et maillage. Ceci rend les mélanges successifs bien plus intuitifs. Nous avons insisté dans le chapitre d'introduction sur l'expérience utilisateur qui nous le pensons est fondamentale pour la modélisation de formes 3D. C'est notre premier essai de création d'une expérience utilisateur aboutie pour la création de formes libres ; et ce prototype nous a permit de montrer que l'expérience utilisateur que l'on désire implique de choisir judicieusement les modèles mathématiques/géométriques que l'on utilise.

2.2 Cahier des charges

Pour concevoir un logiciel de modélisation à destination du grand public, la première question que nous devions nous poser est : comment combiner facilité d'utilisation et richesse d'actions ? Comment créer un logiciel qui s'adapte au plus grand nombre, que tout un chacun sache utiliser de manière instinctive tout en étant robuste ? Pour bien faire il faut essayer de comprendre la représentation qu'une personne non-informaticienne, non-mathématicienne se fait de la modélisation numérique de formes libres :

Perception de la 3D La 3D est souvent pensée comme naturelle, allant de soit. L'utilisateur s'attend à ce que la forme qu'il dessine en 2D sur son écran aille se positionner automatiquement à la profondeur qu'il a en tête. En fait la perception de la 3D sur ordinateur n'est pas évidente du tout, certaines personnes la perçoivent très mal tandis que d'autres n'ont pas conscience que l'écran ne propose qu'une projection 2D d'une scène 3D.

Surface ou Volume ? Une forme peut être pensée comme étant l'un ou l'autre. Bien que ces représentations paraissent équivalentes nous savons bien en pratique que passer de l'une à l'autre n'est pas trivial et nécessite des modèles spécifiques.

Qu'est-ce que la topologie ? Un utilisateur novice tend à créer des formes de genre topologique complexes alors qu'il ne connait même pas la signification de ce mot. Dessiner des formes avec des multitudes de trous et d'anses ne devrait pas poser de difficulté particulière au sein du logiciel.

Petits détails De même il est naturel de vouloir ajouter de petits détails sur un objet, alors que cela est souvent impossible dans les logiciels de modélisation intuitive du à un manque de gestion de la multi-résolution.

Positionnement Cette action pourtant simple peut se révéler laborieuse. Les opérations de zoom, translation, rotation doivent pouvoir se faire instinctivement, alors qu'elles nécessitent pas moins de 6 degrés de liberté. De plus, garder ses repères dans l'espace virtuel est difficile, l'utilisateur peut rapidement se perdre. Le logiciel Teddy, mais il n'est pas le seul, bloque tout simplement le zoom et la translation pour éviter à l'utilisateur de se perdre.

Ces observations ont conditionné le système que nous présentons ici car nous voulions que les aspects que nous venons d'aborder ne dérangent pas l'utilisateur. Dans ce travail nous avons donc tout particulièrement travaillé la gestion de la topologie pour qu'elle soit transparente à tous les niveaux de création, amélioré la perception de la profondeur et rendu plus instinctif l'ajout par croquis de nouvelles primitives. Pour finir nous n'avons pas restreint le modèle à un seul niveau de résolution : grâce à notre représentation

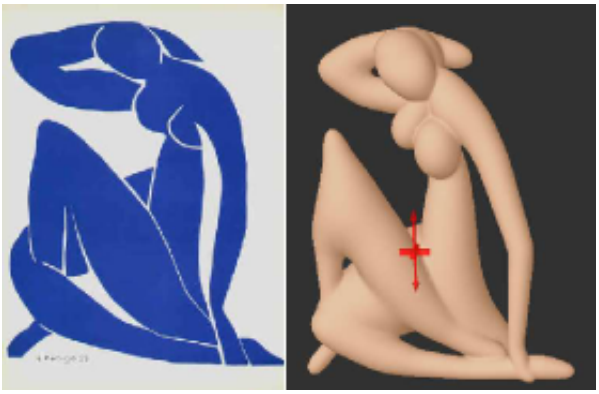

FIGURE 2.1 – À gauche : l'œuvre originale de Matisse. À droite : Un modèle 3D similaire créé avec notre système en 10 étapes depuis différents angles de vues. L'œuvre originale a servi d'inspiration.

de la forme et à une navigation aisée, l'utilisateur peut ajouter de minuscules détails sur une grande surface de manière transparente.

2.3 Vue d'ensemble

Ce travail présente un système interactif de modélisation destiné aux utilisateurs novices. Notre système s'appelle MATISS, et fonctionne sur la base d'assemblages successifs de formes qui se mélangent de manière lisse et automatique. Des formes arbitraires, lisses de topologie quelconque sont créées progressivement en mélangeant des surfaces, créée depuis des angles de vue variables, à des niveaux de zoom variables. Pour créer les morceaux d'objets, nous proposons une interface par croquis, qui, plutôt que d'utiliser une métaphore de type crayon, utilise une interface de type pinceau. Cet aspect est la principale contribution de notre système puisqu'il permet de lever naturellement l'ambigüité des contours dessinés ; ce qui rend naturelle la création de pièces de genre topologique quelconque. Avec MATISS, des formes lisses et arbitraires peuvent être construites progressivement en peignant et en ajoutant des détails depuis plusieurs points de vue. Les principales fonctionnalités de notre système sont :

– Un mécanisme de construction de surface à partir de régions peintes qui ne nécessite aucun pas d'optimisation.

– Une manière améliorée d'inférer la profondeur d'une forme grâce aux éléments déjà dessinés.
– Un nouveau mécanisme de mélange, qui est à la fois lisse, restreint aux zones d'intersections et qui s'adapte naturellement à la taille des composants mélangés.

La figure 2.1 illustre un modèle créé dans notre logiciel.

2.3.1 Peindre des régions plutôt que dessiner des silhouettes

On s'intéresse dans cette section à la création de volumes 3D élémentaires. Pour cela nous avons choisi de ne pas utiliser une approche classique par croquis, qui consisterait à laisser l'utilisateur dessiner les contours de l'objet qu'il souhaite créer. Au lieu de cela, nous laissons l'utilisateur peindre des régions de l'écran comme il le ferait dans Paint^{TM}ou tout logiciel de peinture. Lorsque l'utilisateur est satisfait de sa peinture il peut convertir la peinture en 3d. Par extraction de l'axe médian, un squelette est alors associé à chaque région peinte. Pour générer la forme nous créons une surface implicite que nous maillons finalement, comme nous le verrons dans la section 2.4.4.

Dans la plupart des systèmes dits "<par croquis"> [44, 67, 9, 46], l'utilisateur doit dessiner un contour 2D, duquel est extrait une forme 3D par "<gonflement">. À l'exception de [46], le contour est composé d'une seule ligne fermée, qui est interprétée comme une silhouette plane ; ce qui restreint la famille des formes ainsi créées. Néanmoins des formes complexes peuvent malgré tout être construites à l'aide de ce mécanisme par mélange successif de ces formes élémentaires, dessinées depuis des points de vues différents.

Dans ce travail nous réutilisons l'idée de créer des formes élémentaires qui ont des silhouettes planes ; puisque cela rend la prise en main par les utilisateurs novices plus simple et que cela facilite la reconstruction 3D comparé aux silhouettes complexes, dont la profondeur est variable. Cependant, afin de faciliter la création de formes de topologie arbitraire nous utilisons une métaphore de peinture plutôt que de dessin. L'utilisateur choisi une brosse ainsi que sa taille et peut, librement, peindre un trait, fermé ou non, ou bien remplir une région. Cette dernière opération peut aussi être effectuées aisément à l'aide de l'outil de remplissage "<pot de peinture">. Une gomme

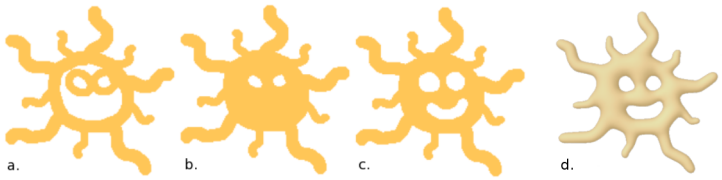

FIGURE 2.2 – Détail des étapes de création d'une peinture de soleil complexe. Noter les fins rayons de soleil ainsi que les trous figurants les yeux et la bouche.

est également disponible dans l'interface : voir la bouche dans l'interface 2.2. Cette approche permet de lever facilement une ambigüité inhérente au dessin : distinguer ce qui est dedans de ce qui est dehors.

Utiliser une métaphore de peinture a plusieurs avantages : D'un point de vue utilisateur les éléments longs et fins peuvent être peints en un seul geste et la correction à l'aide de la gomme est intuitive. Du point de vue du système il n'est pas besoin de surveiller les actions utilisateur pour s'assurer, par exemple, que le contour est bien fermé ou qu'il est correctement orienté, qu'il n'est pas à l'intérieur d'un autre contour, qu'il ne se superpose pas. Par ailleurs aucune interface supplémentaire n'est nécessaire pour créer une forme de genre arbitraire. Des régions complexes telles que celles décrites dans la figure 2.2, sont définies naturellement à l'aide de régions.

2.4 Conversion d'une région peinte en forme 3D

Cette section détaille le processus que nous utilisons pour créer une forme 3D à partir d'une action utilisateur en 2D. Après avoir expliqué les avantages d'utiliser des zones peintes plutôt que des silhouettes, nous détaillons la manière dont nous extrayons un squelette de convolution des régions peintes et dont les paramètres des surfaces de convolution sont choisies de manière à ce que la forme 3D soit ajustée aux régions dessinées.

Cette partie du travail fut plus développée par Adeline Pihuit. En particulier le choix des paramètres que convolution qui est particulièrement bien détaillé dans son manuscrit de thèse et qui fut l'objet d'extensions [61]. Pour cela nous ne nous attardons pas trop sur cette partie.

2.4.1 Conversion des zones peintes en squelettes

Les régions peintes par l'utilisateur sont enregistrées dans une texture de taille fixe pour être traitées ensuite. La deuxième étape est d'extraire l'axe médian des régions peintes. Ce dernier est ensuite converti en graphe de lignes brisées. L'axe médian est par définition le lieu des cercles maximaux contenu dans une région. Une fois le graphe extrait il est traité, simplifié, pour pouvoir être utilisé en tant que squelette de convolution [13].

Plutôt que de calculer une triangulation de Delaunay Contrainte, puis en extraire l'axe médian comme dans [44, 73, 9], nous utilisons une méthode qui permet d'extraire rapidement, par érosion itérative, un ensemble de pixels qui approxime l'axe médian [41]. L'image squelette résultante est enregistrée dans une seconde texture. Grâce à cette méthode la résolution du squelette est fixée par la taille de la texture que nous utilisons, choisie pour correspondre à la résolution de l'application. En pratique les squelettes ont malgré tout des tailles variables une fois projetés en 3D, puisque l'utilisateur peut librement zoomer dans la scène avant de commencer à peindre.

En sus de l'image squelette nous calculons une image de distance, contenant en tout point des zones peintes, la distance au bord le plus proche. Pour cela nous appliquons à l'image une Transformée de Distance Pondérée (en anglais : Weighted Distance Transform) qui utilise un algorithme de balayage des lignes pour propager les distances depuis les bords d'une zone. Les distances sont initialisées à 0 en dehors et 1 à l'intérieur des régions peintes. Deux balayages, sont alors appliqués, dans des sens opposés, appliquant respectivement les opérations f_1 et f_2 aux pixels balayés :

$$f_1(P) = min(N_5 + a, N_4 + b, N_3 + a, N_2 + b) \tag{2.1}$$

$$f_2(P) = min(P, N_1 + a, N_8 + b, N_7 + a, N_6 + b) \tag{2.2}$$

où a et b sont des poids de distance (nous utilisons la métrique $3-4$ de la Figure 2.3 (b)) et $N_{i=1..8}$ sont les voisins du pixel traité. Cet algorithme permet de calculer efficacement pour chaque pixel de l'image une approximation du rayon des disques disques maximaux centrés en ce pixel et compris dans la zone (voir Figure 2.3 (c)).

La dernière étape à accomplir est de transformer ces images, de squelette et

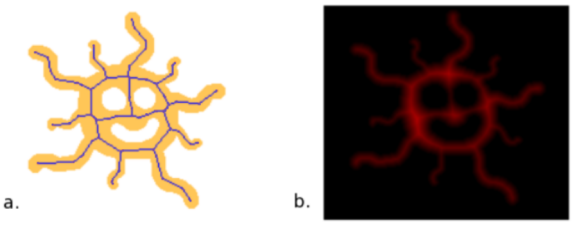

FIGURE 2.3 – A gauche : Image du squelette obtenu par érosion de la zone peinte. A droite : carte de distance, l'intensité de gris indique la distance à la surface.

de distance, en squelette de convolution ; c'est à dire en graphe de branches ayant des poids adaptés à leurs sommets ; de manière à ce que la surface de convolution engendrée s'ajuste aux contours désirés.

Pour commencer le squelette est rééchantilloné et converti en poly-lignes, à une résolution adéquate. Pour être certains que la forme, aussi bien que la variation de rayon le long des branches, soit bien capturée, nous utilisons une résolution d'échantillonage qui dépend de la géométrie et de la taille de la brosse. Nous commençons par créer des sommets aux lieux des points de branchement ; c'est à dire des pixels squelettes ayant plus de 2 voisins. Nous créons aussi des sommets aux extrémités de branche, là où les pixels squelettes n'ont qu'un voisin. Nous raffinons alors les branches en ajoutant des sommets le long des pixels squelettes là où la courbure devient trop importante. Pour finir nous raffinons encore un peu plus les branches si ces dernières s'éloignent trop de la branche squelette équivalente. Plus d'information sur cette partie peuvent être trouvées dans la thèse de Adeline Pihuit. Un résultat de la squelettisation peut-être observé sur la Figure 2.3(b).

Le traitement effectué dans cette partie convertit la texture peinte en graphe de segments de polylignes qui sera ensuite utilisé comme squelette de convolution. Grâce à l'image de distance calculée dans cette étape la valeur r_i, correspondant au rayon des sphères maximales incluses dans la zone, est enregistrée dans chaque sommet du squelette.

2.4.2 Conversion des squelettes en surfaces de convolution

Comme énoncé précédemment, nous supposons que, lors de la reconstruction de la forme 3D à partir de l'image 2D, l'épaisseur de la forme en un point doit être similaire à son épaisseur dans le plan de peinture. Plus la région peinte est large plus l'épaisseur de la forme résultante doit être importante. Les surfaces implicites sont un choix naturel pour la reconstruction d'une surface lisse partant d'un squelette. De plus, cette représentation facilite le mélange des formes de base depuis des points de vue différents. Pour pouvoir utiliser le graphe de polylignes comme squelette implicite tout en évitant le problème bien connu de gonflement des formes au niveau des joints [24], nous nous servons des surfaces de convolution. Pour être précis nous utilisons la formule analytique de la convolution, par noyau de Cauchy, sur des segments à poids variant linéairement, également utilisé dans [73]. La surface implicite est définie comme l'ensemble des points P tel que : $F(P) = T$ où T est une iso-valeur choisie et la fonction de champs F est l'intégrale de la fonction noyau le long du squelette S_k :

$$F(P) = \int_{S_k} h_S(P) w(P) dS \tag{2.3}$$

Puisque l'intégrale sur support donné est égale à la somme des intégrales sur chaque partie d'une partition de ce support, F est calculée en sommant les intégrales le long des différents segments qui composent le squelette. Afin de calculer rapidement et efficacement les champs potentiel nous utilisons un noyau de Cauchy pour le calcul, noyau pour lequel une formule fermée de l'intégrale [69] est :

$$h_S(P) = \frac{1}{(1 + s^2 d^2(P, S))^2} \tag{2.4}$$

où $d(P, S)$ est la distance entre le point P et le point sur squelette S. s est un paramètre qui permet de contrôler la largeur du noyau, et donc la manière dont cette primitive va se mélanger avec les autres.

La solution de l'intégrale du noyau de Cauchy, originellement définie sur un segment droit de poids constant, peut être étendue [73], aux segments droits

de poids variant linéairement. En utilisant la formulation donnée dans [73], nous exprimons le champs potentiel au point P comme :

$$F(P) = w_H F_H(P) + \frac{w_T - w_H}{l} F_T(P) \tag{2.5}$$

où w_H et w_T sont respectivement les poids des sommets de tête et de queue d'un segment. De plus $F_H(P)$ et $F_T(P)$ sont des fonctions de champs qui ne dépendent que de la tête et de la queue d'un segment. l quant à lui, est la longueur du segment.

2.4.3 Choix des paramètres de convolution

On s'attendrait intuitivement à ce qu'utiliser les rayons que l'on désire comme poids au sommets des segments du squelette, permette de reconstruire une forme d'épaisseur adéquate. Malheureusement un réglage supplémentaire est nécessaire dû à la forme du noyau de convolution utilisé. La forme de la fonction de champs générée par une convolution ne ressemble pas à un champ de distance, elle dépend de la forme du noyau utilisé pour la convolution. Des travaux précédents ont résolu ce problème soit en enregistrant les associations poids épaisseur dans un tableau puis en inversant ce tableau [8] ou en effectuant une optimisation itérative pour trouver les valeurs de poids qui produisent la surface la mieux ajustée. Au lieu de cela nous utilisons une formule directe pour calculer les poids w_i :

$$w_i = (C \times r_i)^3 \tag{2.6}$$

où r_i est le rayon désiré au niveau du sommet d'une branche, et C est un facteur constant dépendant de la taille de l'image, servant seulement à garder le potentiel dans des bornes raisonnables. Les longueurs des rayons r_i étant exprimées en pixels, nous utilisons :

$$C = (\texttt{(taille (en pixels) de l image)}^{-1} \tag{2.7}$$

La puissance 3 que l'on peut observer dans l'équation s'explique par le noyau de convolution utilisé. En effet, selon le noyau utilisé, la fonction de champs

décroit différemment en s'éloignant du segment de convolution. La puissance 3 correspond à l'inverse du comportement asymptotique de l'intégrale du noyau de convolution en s'éloignant d'un segment.

Pour mélanger facilement des champs potentiels de surfaces implicites il faut que ces surfaces aient la même iso-valeur. La dernière étape consiste donc à re-normaliser notre champ potentiel. Pour cela nous commençons par chercher l'iso-valeur qui s'ajuste le mieux à la peinture. Nous calculons la fonction de champs en plusieurs points du contour sur lequel on souhaite ajuster la surface implicite. La moyennes des valeurs relevées à proximité des extrémités de chaque branche nous donne l'iso-valeur de notre surface implicite. Pour re-normaliser nous multiplions le champs par l'inverse de l'iso-valeur trouvée.

La fonction de convolution résultant de cette étape est présentée en Figure 2.3 (c). La surface ainsi générée est bien lisse mais ne fait qu'approximer la région peinte. [61] présente une méthode qui permet d'améliorer les poids w_i sans passer par une méthode d'optimisation. Dans un certain sens cette représentation de convolution n'est pas très adaptée à la modélisation par squelette car les poids w_i ne sont pas corrélés suffisamment au rayon de la forme générée. La thèse en cours de Cédric Zanni, aborde ce sujet.

2.4.4 Maillage d'un élément de surface

La surface de convolution générée précédemment est maillée à l'aide d'une méthode classique de marching cube [21]. Pour ne pas perdre de petits détails nous choisissons la résolution de la grille de maillage comme la moitié de la taille du plus petit détail du squelette de convolution de manière à capturer tous les détails. La taille du plus petit détail nous est simplement donnée par le plus petit des rayons des sommets du squelette de convolution de la primitive que l'on fusionne.

Pour rendre les calculs de potentiel plus efficaces nous tronquons la fonction de champs de chaque segment de convolution au delà d'une boite englobante centrée sur le segment. Cela est possible car le noyau de Cauchy, bien que non borné, décroit rapidement vers zéro en s'éloignant de son centre. Dans notre implémentation nous utilisons une boite englobante alignée sur les axes principaux du monde qui englobe le segment ainsi qu'un rayon supplémentaire correspondant au maximum des r_i des sommets du segment. La taille de

cette boite est enfin multipliée par un facteur constant, 3 dans notre cas. La fonction de champs associée à ce segment est alors considérée comme nulle en dehors de cette boite englobante.

2.5 Positionnement des nouvelles formes

Dans le système MATISS, des primitives 3D peintes depuis des points de vue différents sont utilisées pour générer une forme 3D. Quand quelqu'un peint ou dessine sur une feuille de papier c'est son imagination qui choisi la profondeur relative des éléments qui composent l'œuvre. Il imagine alors la troisième dimension associée à son dessin. Dans un système informatique la profondeur doit être explicitement calculée. Pour cela nous essayons d'inférer, autant que possible, la profondeur relative des éléments d'une scène ; tout du moins nous souhaitons rendre ce choix, fait par le système, le plus intuitif possible.

2.5.1 Peindre de front à la bonne profondeur

L'idée intuitive est de peindre dans un plan perpendiculaire à la direction de la caméra. Quand l'utilisateur clique sur la scène à l'aide de l'outil peinture une toile de peinture (un canevas) apparait.

La stratégie que nous utilisons pour choisir la profondeur de ce plan est, par défaut, de garder la profondeur à cette toile constante. Par contre, lorsque l'utilisateur à déjà commencé à peindre une forme, il veut le plus souvent lui adjoindre d'autres parties ou détails depuis un point de vue différent. Nous utilisons alors un mécanisme de positionnement relatif : quand la peinture commence sur un objet déjà existant, nous créons un plan frontal à la profondeur du point où l'utilisateur a cliqué, de manière similaire à [31]. Puisque le squelette du nouvel élément de surface est situé sur ce plan, les formes se chevauchent significativement, ce qui permet de les mélanger de manière lisse. Le mélange lisse est expliqué dans la section 2.5.2.

Pour améliorer le retour visuel pendant que l'utilisateur explore la scène nous affichons en permanence un plan frontal, quasi transparent, qui indique à quelle profondeur le prochain dessin se situera si l'utilisateur clique à cette

FIGURE 2.4 – Ajout d'un nez à une figurine en forme d'étoile. L'utilisateur dessine le nez dans un plan transparent qui coupe l'étoile à la profondeur désirée (à gauche) puis le nez est créé lorsque la caméra change de position (à droite).

position. Ce plan virtuel est affiché de façon à ressembler à une feuille de calque qui viendrait couper la scène en deux. La partie de l'objet qui se trouve devant le plan parait identique alors que la partie arrière se trouve être éclaircie (Figure 2.4). Quelque soit le plan de peinture, la forme est créée dans l'espace du plan de peinture puis positionnée en 3D à la bonne échelle et profondeur de manière à ce que la forme et la région peinte se superposent.

Création d'un pont entre des formes existantes

Dans certains cas, l'utilisateur souhaite peindre une connexion entre plusieurs éléments déjà existant d'une scène. C'est par exemple le cas quand on souhaite dessiner une tige reliant deux objets entre eux. Dans ce cas on ne s'attend pas à peindre une zone perpendiculaire à la caméra mais plutôt un plan oblique qui connecte ces éléments. Notre système autorise un tel mécanisme, dit de création de ponts (Illustré sur la figure 2.5). On peut utiliser ce mécanisme pour relier deux objets différents ou bien deux points d'une même forme sans avoir à repositionner la caméra.

L'utilisateur choisi deux points en cliquant à l'écran. La profondeur de chacun des points est calculée pour que l'objet soit positionné au milieu de la forme qu'il pénètre. Pour cela nous choisissons la profondeur à mi distance de la face avant et de la face arrière à l'endroit ou l'utilisateur a cliqué. Puisque deux points ne suffisent pas à définir un plan, nous choisissons de prendre le plan qui est vertical, au regard de la position actuelle de la caméra. Quand l'utilisateur peint sur un plan oblique, sa peinture est distordue de manière à coïncider avec la forme que l'on souhaite créer. Pour garder le bénéfice d'avoir une image de peinture de taille constante nous continuons à calculer

FIGURE 2.5 –

le squelette de convolution comme si le plan était face à la caméra. Nous projetons alors le squelette dans le plan oblique et nous adaptons les rayons aux sommets du squelette en conséquence. Le squelette oblique résultant génère la surface de convolution.

2.5.2 Mélange lisse des composants

Après la reconstruction, la où, les formes nouvellement créées sont mélangées aux objets déjà présents dans la scène. Puisque nos objets sont construits en utilisant des surfaces de convolution dont le champs a été tronqué à zéro à l'extérieur d'une certaine boite englobante, ce sont toujours des surfaces implicites à support compact. Nous pouvons donc utiliser tous les opérateurs de mélange implicite s'appliquant à ces fonctions de champ. Cela va de la simple somme [20, 23] à des opérateurs plus avancés tel que l'opérateur arc-d'une-ellipse [16] ou de déplacement [43]. Notre champ global n'est pas C^0 partout mais l'opérateur arc-d'une ellipse à ceci de bien qu'il peut le garder C^1 partout à proximité de la surface.

Puisque notre système autorise l'utilisateur, de manière transparente dans la même interface, à créer aussi bien de gros éléments que de très petits, nous devons choisir un opérateur de mélange qui surmonte le problème connu de l'absorption des petits détails par les grands [82]. Ce problème est décrit dans la section 1.1.1 de l'état de l'art. Pour le résoudre nous utilisons la seconde version de l'opérateur arc-d'une-ellipse U_B, équation 2.8, qui permet de contrôler précisément le mélange indépendamment de la taille de chaque objet. Cette version est particulièrement adaptée à notre situation puisqu'elle permet de contrôler la forme du mélange de deux champs potentiels de décroissance différentes.

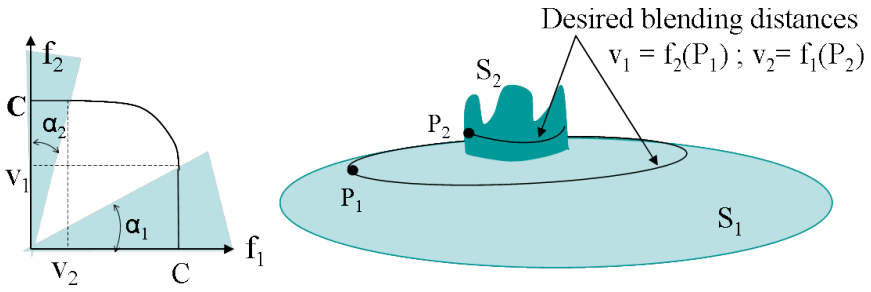

FIGURE 2.6 – Pour localiser le mélange à l'aide de l'opérateur de Barthe il suffit de choisir deux points $P'1$ et $P'2$ qui vont restreindre le mélange.

$$U_B(Xp, Yp) = \begin{cases} X_p & \text{si } Y_p = 0 \\ Y_p & \text{si } X_p = 0 \\ X_p & \text{si } \theta_p \leq \theta_1 \\ Y_p & \text{si } \theta_p \geq \theta_2 \\ C_p, \text{ solution de :} & \\ \quad \frac{(X_p - C_p.cot(\theta_2))^2}{(C_p - C_p.cot(\theta_2))^2} + \frac{(Y_p - C_p.tan(\theta_1))^2}{(C_p - C_p.tan(\theta_1))^2} = 1 & \text{si } \theta_1 < \theta_p < \theta_2) \end{cases} \tag{2.8}$$

Soit F_1 et F_2 les champs potentiels définissants une paire d'objets à combiner, O_1 et O_2. L'opérateur que nous utilisons restreint le mélange à un volume, apriori non borné, localisé entre deux iso-valeurs : V_1 et V_2 les valeurs respectives des fonctions F_1 et F_2 prises à la frontière du mélange sur respectivement O_2 et O_1 (Voir Figure 2.6). Cet opérateur nous permet justement de choisir ces deux iso-valeurs et ainsi d'obtenir un mélange lisse tout en évitant de trop lisser la forme plus petite. Pour choisir V_1, nous évaluons F_1 en un point P'_1 qui se trouve à une distance adéquate de l'objet O_1 le long du gradient du champs F_1.

De cette manière nous avons $V_1 = F_1(P'1)$ et nous choisissons :

$$P'_1 = P_1 - w_{2,min} \frac{\nabla F_1(P_1)}{\|\nabla F_1(P_1)\|} \tag{2.9}$$

Avec P_1 un point choisi sur O_1 proche de l'intersection et $w_{2,min}$ le minimum des poids des sommets du squelette de O_2. $w_{2,min}$ donne une bonne estimation

du plus petit détail de O_2. Finalement nous transposons cette démarche de manière symétrique pour calculer V_2.

Pour choisir P_1 nous utilisons les représentations explicites que nous avons des objets O_1 et O_2, c'est à dire les maillages M_1 et M_2. Nous calculons l'union des maillages M_1 et M_2, puis utilisons les points de l'intersection pour choisir P_1 et P_2.

Nous expliquons la méthode de maillage dans la section suivante. Par chance l'union des maillages est une étape de cette méthode et ne pose donc pas un surcout particulier.

2.5.3 Application locale du mélange

Un problème bien connu de la modélisation implicite est le mélange non-désirable. Ce problème est d'autant plus embêtant dans un système de modélisation car l'utilisateur ne connait pas les surfaces implicites. Ce qu'il souhaite c'est que les objets qu'il assemble se mélangent là ou ils se superposent. Nous proposons ici, un opérateur hybride de mélange local, qui nous permet de mettre en place une solution intuitive plus simple que celles décrites dans l'état de l'art (section 1.1.1).

Avant tout nous considérons que l'utilisateur à exprimé son souhait de mélanger certaines régions en les faisant se recouvrir (Figure 2.7). On s'attend donc à ce que le mélange s'effectue à proximité de l'intersection des maillages. L'idée est donc de sélectionner la zone du maillage à recalculer autour de l'intersection des maillages, et de n'appliquer le mélange que sur la zone ainsi définie. Nous commençons par calculer l'union M_U des maillages grossiers M_1 et M_2, en même temps que le maillage de leurs contours d'intersection M_C. Les contours d'intersection sont des courbes ; modélisées par des lignes brisées fermées ; qui peuvent être multiples. Dans notre implémentation nous utilisons la bibliothèque GNU GTS sous licence LGPL pour ces calculs.

Nous commençons par soustraire les sommets de l'union M_U qui doivent être recalculés pour les ajouter à la zone d'intersection M_I. Pour cela nous commençons par ajouter M_C à M_I puis nous procédons par propagation, commençant ainsi par le voisinage des contours d'intersection M_C. Au cours de la propagation nous traitons un à un les sommets P du maillage M_U connexes à M_I. Si $F_1(P) > V_1$ et $F_2(P) > V_2$ alors le sommet appartient à

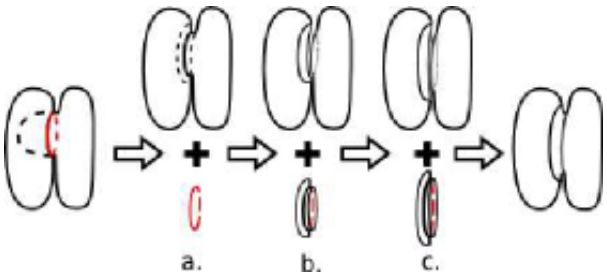

FIGURE 2.7 – Pour appliquer localement le mélange calculons l'union des maillages puis propageons le maillage à proximité de la courbe d'intersection.

la région qui doit être recalculée, il est enlevé de M_U et ajouté à M_I. Si P est recalculé alors ses voisins non traités sont ajoutés à une liste pour être traités à leur tour. Les triangles ayant trois sommets à recalculer sont alors déplacés de M_U vers M_I. Le maillage M_u est correcte alors que le maillage M_i, correspond à l'intersection qui doit être remaillée. Pour remailler, nous commençons pas décimer M_I puis nous appliquons un processus de subdivision adaptative en utilisant un critère de courbure [26]. Une fois M_I remaillé nous le réincorporons à M_U, en évitant bien-sûr de subdiviser les arrêtes communes à M_U. De cette manière nous avons localement remaillé la partie du maillage se situant au niveau de l'intersection en évitant les problèmes de recollement et en évitant de recalculer les parties du maillages non affectées par le mélange. Chaque point créé lors de la subdivision est immédiatement déplacé, dans la direction du gradient du champs potentiel, de manière à se trouver sur la surface implicite.

Un avantage de cette méthode est que bien que nous n'utilisons pas de graphe de mélange, le mélange reste local. Grâce à l'étape de propagation, le remaillage ne s'effectue que dans les zones connexes à l'intersection. Reprenons l'exemple du personnage auquel on rajoute un bras. Quand bien même l'extrémité du bras se trouve dans une zone dont la surface implicite associée est modifiée par le mélange, si la zone à remailler ne s'est pas propagée jusqu'à l'extrémité du bras, alors il restera inchangé et ne se mélangera pas au corps. Si d'autres parties de ces deux surfaces se rapprochent, il n'y aura ni déformation, ni intersection. Ce mécanisme rend le comportement du mélange plus prévisible que dans les outils de modélisation implicites usuels. Une limitation est que le maillage ne correspond plus à la fonction de champ potentiel globale de l'objet tel que définie habituellement. Chaque élément

de maillage est plutôt associé à une sous partie du graphe de potentiel. Fusionner une primitive à ces endroits peut alors déclencher les mélanges qui avaient été stoppés. Cette limitation est abordé et une solution beaucoup plus générale pour permettre la localité du mélange est développée dans le chapitre suivant.

Cette méthode ne garanti pas non plus que la surface implicite a une topologie égale à celle de la surface maillée, en particulier au niveau de la zone d'intersection. La surface maillée ne pouvant changer de topologie, le maillage résultant est alors de piètre qualité, bien que l'algorihtme de maillage reste stable.

2.6 Résultats et Performances

Le système MATISS fournit à l'utilisateur un ensemble d'outils pour peindre, générer des formes, explorer le modèle, mais aussi des fonctionnalités traditionnelles de sauvegarde, restauration, annuler et refaire.

Pour la partie peinture l'utilisateur à accès à des outils classiques de peinture : des brosses, un pot de peinture, une gomme. Une palette de peinture est aussi affichée et il suffit de cliquer sur une couleur pour qu'elle devienne la couleur active. Enfin deux boutons sont disponibles pour valider un dessin et lancer la reconstruction ; ou annuler la reconstruction et revenir à l'état d'exploration de la scène (Figure 2.9).

Dans la vie de tous les jours, quiconque dessine ou modélise une nouvelle forme a tendance à vouloir tourner autour de sa création pour en valider la qualité. Ce réflexe, normal, vient du besoin de valider et d'analyser une action pour, soit la corriger, soit envisager l'étape suivante de la création. Pour que cette action de validation reste instinctive nous avons rendu la validation du dessin optionnelle. Ainsi la forme 3D est automatiquement générée et affichée quand l'utilisateur change de point de vue. Puisque qu'un simple clic bien placé permet de créer le plan de dessin ; l'utilisateur ne se focalise durant l'édition que sur, alternativement, la peinture et l'observation de sa création. Peindre de nouveau et observer de nouveau. L'interface qui découle de ce fonctionnement peut être utilisée directement par le plus grand nombre avec un besoin d'apprentissage minimal. La même interface est utilisée pour créer

le modèle, de la première à la dernière pièce.

Le système de positionnement a été particulièrement travaillé pour pouvoir être pris en main immédiatement. C'est une fonctionnalité élémentaire mais très importante. Selon nous, un bon système de positionnement pour novice, doit être simple à comprendre et intuitif et efficace. La modélisation se fait par rapport à un objet ; le déplacement doit donc se faire relativement à l'objet, non dans le vide. Pour décaler un objet il suffit de le saisir et de le déplacer, l'objet saisi reste alors en dessous du curseur de la souris. Pour le Zoom c'est là aussi la partie de l'objet se trouvant en dessous du curseur qui sert de mire de zoom ; un zoom avant ou arrière ce fait alors relativement à ce point. Tirer la molette de la souris vers soi rapproche l'objet de la caméra. Pour la rotation, c'est encore le point de l'objet cliqué qui sert de pivot à la rotation. Pour traiter les rotations nous aurions pu utiliser une track-ball. Nous avons choisi d'utiliser des angles d'Euler en bloquant le dernier degré de rotation. Cette méthode ne nécessite aucun temps d'apprentissage et surtout ne modifie le tangage de la caméra (la verticale). Dans le monde réel la pesanteur indique en permanence la verticale à notre oreille interne, même quand nous penchons la tête. Ce n'est pas le cas dans le monde virtuel et c'est pour cela qu'avoir une caméra qui ne tangue pas est essentiel.

Ce soin porté au positionnement à permis de toucher une population large d'utilisateurs lors des tests que nous avons effectués.

2.6.1 Performances

Dans MATISS l'ajout d'un nouvel élément de surface nécessite 3 étapes principales : le calcul du squelette de convolution, le maillage initial de l'élément et enfin le mélange de ce maillage avec les éléments de la scène. La Table 2.1 donne quelques temps de calcul sur un ordinateur muni d'un processeur Centrino ™1.8 Ghz. Les temps sont donnés en millisecondes. Notez que le mailleur que nous utilisons dans la section 2.5.3 n'est pas optimisé. Les résultats seraient bien plus rapide en améliorant cette étape. Ces temps n'incluent pas les améliorations faites dans le chapitre suivant sur le mélange local de surfaces implicites.

Modèle	Calcul Squelette	Maillage	Mélange
Soleil	120	6500	-
Étoile : corps	70	1180	-
Arbres : troncs	30	1040	-
Arbres : branches	100	5360	750
Arbres : pont	20	310	740

TABLE 2.1 – Temps de calcul (en ms) pour la génération du soleil (Fig 2.2), de l'étoile (Fig 2.4, des arbres avec le pont (Fig 2.5)

2.6.2 Validation informelle par des utilisateurs

Puisque le système MATISS a été conçus pour le grand public nous avons testé ce système avec deux types d'utilisateurs : des utilisateurs novices ainsi qu'une artiste traditionnelle, Virginia Alphonso, peu familière des outils de création numériques. Les utilisateurs novices furent nombreux puisque nous eûmes, entre autre, l'occasion de présenter le logiciel devant des classes de primaires, à la fête de la science ainsi qu'à la biennale art et sciences de Grenoble. Ces personnes ont pu utiliser le logiciel parfois avec une souris, parfois avec une tablette graphique. Les premiers utilisateurs déplorèrent le manque de couleur car la première version de MATISS en était dépourvue. Les utilisateurs ont apprécié la simplicité de l'outil ainsi que la possibilité de créer une forme 3D, ce qui était une première fois pour la majorité d'entre eux. Les adolescents ont très vite pris en main la navigation dans la scène et la possibilité de zoomer pour ajouter des petits détails. Ils se sont rarement plaint de la forme des éléments de surface reconstruits bien que ces derniers ne soit pas exactement ajustés à leur peinture initiale. À cause de ces formes approximatives les utilisateurs devaient parfois mélanger plusieurs parties différentes pour palier le problème de variation importante des rayons des squelettes de convolution. La figure 2.8 illustre ce problème et la manière dont les utilisateurs l'évitent. La gomme 3D ainsi que la possibilité de déplacer des morceaux de forme furent ajoutés à des versions ultérieures du logiciel. L'enthousiasme généré lors des dernières présentations de MATISS nous a montré que nous avons atteint notre objectif en terme d'interface grand public.

La seconde catégorie d'utilisateur sont des artistes ; qui ont utilisé le système à l'aide d'une tablette avec écran intégré. L'un est familier de la modélisation

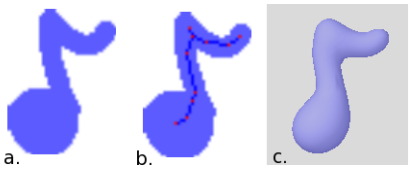

FIGURE 2.8 – Le contrôle de la courbe à partir d'un unique dessin est parfois complexe avec Matiss ; en particulier dans les zones ou le rayon du squelette varie rapidement et dans les concavités.

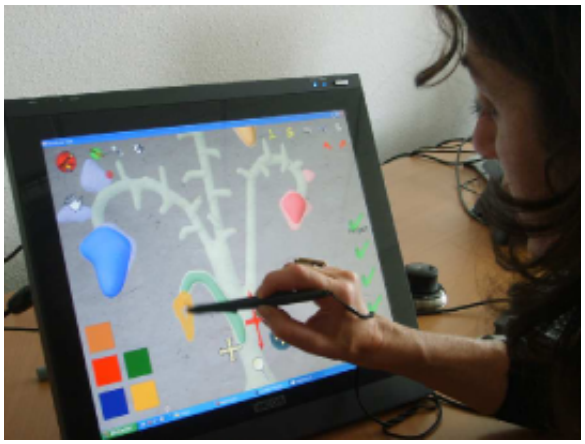

FIGURE 2.9 – Un artiste traditionnel qui utilise MATISS

numérique alors que l'autre n'avait jamais travaillé avec un ordinateur avant. L'artiste traditionnel a pris le logiciel en main très rapidement et a apprécié les fonctionnalités communes tel que l'enregistrement, la chargement de fichier, l'annulation et la restauration des actions. Tous deux ont demandé une reconstruction plus précise de leur peintures. Ils ont aussi demandé un contrôle plus fin de la courbure des formes créées, entre autre pour créer des formes plates telles que des feuilles ou des pétales (Fig 2.9). Dans l'ensemble les deux ont aimé la simplicité de MATISS pour créer des formes 3D et ont demandé à revenir pour d'autres sessions de test.

2.7 Conclusion

Dans ce travail nous avons présenté un système de modélisation interactive de formes volumiques conçu pour des utilisateurs novices de la modélisation numérique de formes 3D. Ce logiciel permet la création et le raffinement de formes lisses en créant des éléments à partir de peintures depuis des angles de vue différents, à des échelles différentes. Les éléments de surface créés par peinture sont représentés de manière implicite en utilisant des surfaces de convolution. De même, ces éléments de surface sont mélangés à l'aide d'opérateurs de mélange de surfaces implicites. Pour représenter ces surfaces nous utilisons des maillages triangulaires qui restreignent la propagation du mélange. Nos résultats montrent qu'il est souhaitable de pouvoir créer des formes de topologie arbitraire de manière transparente, que ce soit durant l'étape de peinture, que durant l'étape de mélange des éléments de surface. L'interface que nous proposons, fondée sur la métaphore de peinture, s'est révélée immédiatement utilisable par des utilisateurs non préparés et souvent inexpérimentés.

La possibilité de supprimer une partie existante d'un modèle, demandée de manière répétée par les utilisateurs, à été implémentée dans une version ultérieure du logiciel, par creusement ou suppression complète de la partie existante. Nous avons réfléchi à des manières plus élaborées de placer les primitives en 3D. L'une d'elle était de projeter chaque élément de squelette sur la surface sous-jacente. Nous avons cherché une manière générique d'inclure un contrôle supplémentaire de la profondeur de la forme sans alourdir l'interface mais n'y sommes pas parvenu, nous voulions proposer une méthode plus naturelle encore que celle proposée dans Shapeshop [67].

Nous souhaiterions aussi améliorer l'adéquation de la forme reconstruite avec la région peinte. Nous voudrions trouver une méthode purement géométrique car une étape d'optimisation alourdirait ce calcul et offrirait moins de contrôle sur le caractère lisse de la surface. Finalement, dans la version actuelle du logiciel, l'épaisseur de chaque élément reconstruit est lié à sa taille locale, sans pouvoir être modifié ultérieurement. Des squelettes composés de parties planes pourraient être utilisés [8] pour étendre les formes possibles des éléments. Ces points sont actuellement repris dans la thèse de Cédric Zanni, actuellement préparée au sein de l'équipe Imagine.

Encore une fois trouver une interface intuitive, et efficace pour intégrer de

telles fonctionnalités est un défi. Ce travail a enfin été validé par un partenaire industriel, Axiatec, se proposant de rendre réelles les créations numériques à l'aide d'imprimantes 3D de qualité.

Chapitre 3

Mélange local de surfaces implicites

3.1 Introduction

Comme nous l'avons vu dans le chapitre précédent le système MATISS propose une expérience utilisateur originale et intuitive. Nous avons cherché dans MATISS à proposer un modeleur qui soit prévisible ; c'est pourquoi MATISS s'appuie sur un mélange local de formes. Cependant, ce mélange est effectué grâce à une solution hybride s'appuyant sur la combinaison de maillages et de champs implicites locaux. La forme résultante ne possède plus de champ potentiel global. Cette partie traite de l'extension de ce type de mélanges aux surfaces implicites classiques. Nous souhaitions proposer des surfaces implicites plus adaptées à la modélisation. C'est pourquoi en plus de traiter le mélange local de surfaces implicites ; notre solution résout plusieurs problèmes d'effacement des petits détails ainsi que de mélange à distance.

L'idée qui inspire la technique que nous proposons est que le mélange ne devrait se faire qu'à proximité des zones où les objets se chevauchent. Les surfaces implicites ne doivent pas exercer d'influence réciproque ailleurs qu'à ces endroits même si elle se rapprochent fortement, comme illustré sur la figure 3.1. A l'opposé, en dehors de cette zone les surfaces devraient rester inchangées, même si la distance qui sépare les deux surfaces est très faible. La méthode que nous proposons s'applique génériquement aux différents types

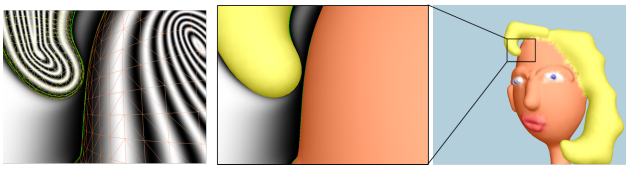

FIGURE 3.1 – Union propre de deux champs potentiels : On peut observer sur cette figure que sans l'opérateur d'union propre la mèche de cheveux se serait mélangée avec le front. Le champ potentiel reste continu dérivable.

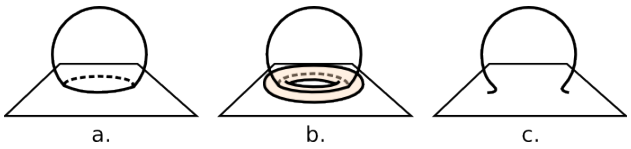

FIGURE 3.2 – Le mélange est localisé dans une zone calculée autour des régions qui s'intersectent.

de surfaces implicites telles que les F-rep, Soft-objects.

Notre solution publiée dans [1] rend l'assemblage de surfaces implicites beaucoup plus prévisible pour l'utilisateur car les surfaces à assembler se mélangent quand elles s'intersectent, ne gonflent plus à distance, et les détails fins ne sont plus lissés.

3.1.1 Contributions

Dans cette méthode nous localisons le mélange automatiquement à proximité des intersections. Pour cela l'idée maitresse de notre travail est de calculer les courbes d'intersections (il peut y en avoir plusieurs) entre les deux surfaces à fusionner et de les utiliser comme squelettes pour définir les régions où les surfaces peuvent se mélanger (voir Figure 3.2). L'épaisseur de la zone de mélange est calculée pour que le mélange croisse progressivement au cours des animations, ce qui permet de modéliser une goute se fusionnant progressivement avec un volume de liquide.

Pour obtenir ce résultat nos contributions sont de résoudre les trois difficultés suivantes :

– Premièrement, nous proposons une méthode efficace mais relativement pré-

cise pour calculer les courbes d'intersection entre deux objets implicites, ainsi que le volume autour d'elles qui définit la zone de mélange.
- Deuxièmement, nous avons besoin d'un opérateur implicite d'union efficace que nous appliquerons partout en dehors du volume de mélange.
- Finalement, nous proposons une fonction et un calcul automatique de paramètres pour traiter le mélange lisse qui se produit à l'intérieur du volume. Notre solution empêche l'atténuation des petits détails au moment du mélange avec un volume plus grand. De manière similaire il empêche aussi l'apparition de gonflement là ou les surfaces sont presque parallèles.

Les sections 3.2 à 3.4 exposent respectivement nos solutions à ces problèmes. La section 3.5 illustre nos resultats sur des exemples simples d'animation ainsi que dans un système de modélisation par croquis étendant le logiciel MATISS du chapitre 2. Finalement nous discutons des forces et des faiblesses de notre approche.

3.2 Volumes de mélange localisés à proximité des intersections

Au lieu de demander à l'utilisateur de sélectionner manuellement les paires de volumes devant se mélanger, nous souhaitons les mélanger automatiquement là où elles s'intersectent et les combiner en utilisant une union partout ailleurs. Pour cela des volumes de mélange doivent être définis automatiquement autour des intersections. Puisque les valeurs de champs du volume de mélange seront utilisées pour interpoler entre l'opérateur de mélange et celui d'union, le volume de mélange doit être une primitive implicite à support local, c'est à dire dont le champ s'annule aux bords de la primitive.

L'intersection entre deux surfaces fermées qui se pénètrent est constituée d'une ou plusieurs courbes fermées. L'idée est d'utiliser les courbes d'intersection comme squelette pour construire un volume de mélange tubulaire qui contiendra le mélange. Dans la section suivante nous discutons de l'extraction efficace d'intersection implicite. Nous expliquons alors comment le volume de mélange est calculé pour offrir un mélange progressif et intuitif quand des objets animés rentrent en contact.

3.2.1 Extraction des courbes d'intersection

Les courbes d'intersection entre surfaces implicites peuvent être décrites de manière analytique (Voir l'Équation 3.1) ou alors discrétisées, par exemple sous la forme d'une liste de poly-lignes. La discrétisation est une approximation de l'intersection mais a l'avantage d'être explicite et donc permet des calculs efficaces. L'expression analytique, d'un autre coté est l'équation exacte de la courbe ou de l'ensemble des courbes fermées, mais donnée sous forme implicite.

$$C_{inter} = \{P \in \mathbb{R}^3 | f_1(P) = C \text{ et } f_2(P) = C\} \tag{3.1}$$

Dans l'équation 3.1 les fonctions f_1 et f_2 désignent respectivement les fonctions de champ des deux objets mélangés et C est iso-valeur pour laquelle les surfaces implicites sont calculées.

Le but est d'utiliser les courbes d'intersection comme squelettes de primitives implicites, définissant les régions de mélange entre les objets donnés en entrée. Fournir une représentation permettant des calculs efficaces de distance entre un point quelconque de l'espace et les courbes d'intersections est donc indispensable. Dans [52], le calcul des courbes d'intersections est effectué sur une représentation discrète du champs dans une grille. Pour les surfaces implicites fonctionnelles une telle représentation n'est pas directement disponible ni souhaitable. Dans le cas discret un calcul de distance peut-être rendu efficace en plongeant la poly-ligne dans une hiérarchie de boites englobantes, mais la géométrie étant statique la précision peut manquer. A l'opposé, si l'on conserve une représentation analytique de la courbe d'intersection, la précision des calculs de distance peut être adaptée aux besoins, mais ces derniers nécessitent d'utiliser des méthodes numériques qui peuvent être coûteuses en temps de calcul.

Pour obtenir précision et efficacité nous utilisions une approche hybride : nous commençons par extraire une courbe d'intersection de basse résolution puis nous la rafinnons localement en subdivisant les segments dont le centre ne respecte pas, étant donné une erreur ϵ, l'équation 3.1. Les points nouvellement créés migrent sur l'intersection en utilisant l'algorithme décrit dans [81], i.e. en convergeant alternativement dans la direction des gradients des deux fonctions potentielles.

L'extraction de la courbe de basse résolution peut-être faite de deux manières différentes. La première est d'utiliser les maillages qui servent à l'affichage de surfaces implicites. L'intersection de ces maillages peut-être calculée efficacement si les maillages sont plongés dans des structures de partitionnement de l'espace tels que des octrees. C'est la méthode que nous avons utilisée dans le logiciel de modélisation. La seconde approche utilise l'algorithme du marching-face présenté dans [50] ; un traitement analogue au marching cube est effectué dans les voxels coupant simultanément les deux surfaces implicites. L'algorithme calcule sur les faces des voxels concernés les points où les iso-surfaces s'intersectent puis relie les points d'intersection les uns aux autres. Cette seconde méthode est plus générale car elle ne nécessite pas d'avoir préalablement maillé les surfaces à intersecter. Tout comme pour le marching cube il peut être couteux de parcourir tous les voxels pour déterminer ceux susceptibles de contenir des intersection. Si l'une des surfaces à déjà été maillée on peut réduire la complexité en ne parcourant que les voxels comportant une portion de maillage. Parallèlement, cette étape peut être optimisée en utilisant l'arithmétique d'intervalles associée à un octree adaptatif englobant le volume d'intersection. La subdivision de l'octree est alors guidée par des fonctions d'inclusion de f_1 et f_2, comme expliqué dans [36, 33, 70], qui fournissent des intervalles quasi-optimaux. Nous avons aussi testé cette approche mais ne l'avons pas utilisée car nous disposions déjà d'une surface maillée.

Au final notre méthode produit un calcul rapide et explicite de courbes d'intersection. Dans le cas de courbes d'intersection multiples nous pouvons traiter individuellement chaque courbe alors qu'un traitement analytique ne le permettrait pas ; ce qui nous permet de générer des volumes de mélange de taille adaptée à chaque courbe.

3.2.2 Choix des paramètres du volume de mélange

A ce niveau de l'algorithme, l'intersection est composée d'une ou plusieurs courbes fermées. Nous construisons un volume de mélange implicite autour de chacune d'elles grâce à une fonction de champs à support compact générée par des segments. Dans notre implémentation nous utilisons les soft-objects de Wyvill (voir section 1.1.1 de l'état de l'art). Malgré tout il reste un paramètre à choisir pour définir notre volume : l'épaisseur du tube qui englobe le volume

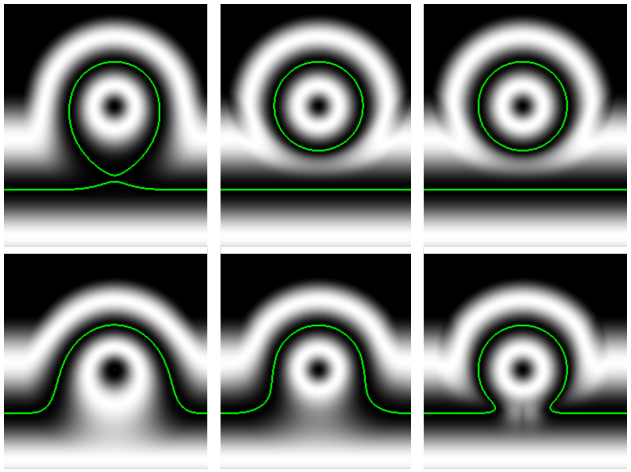

FIGURE 3.3 – Champ potentiel obtenu lors du mélange d'une goute et d'une flaque d'eau implicites. La ligne verte représente l'iso-surface. Sur la ligne du haut la goute est encore à distance, alors qu'elle vient de rentrer au contact sur la ligne du bas. Un opérateur traditionnel (colonne de gauche) provoque un mélange à distance. Au milieu utilise notre opérateur avec un mélange soudain tandis qu'il y a mélange progressif à droite.

de mélange, qui correspond aussi à la taille du support de la fonction de champ associée.

Comme expliqué plus tôt, nous sommes à la recherche d'une méthode qui rende le mélange intuitif dans le cas de la modélisation et aussi de l'animation. En particulier deux surfaces implicites qui rentrent en contact doivent se mélanger progressivement plutôt que de sauter directement à une forme complètement déformée (voir Figure 3.3).

Cette fonctionnalité peut être facilement exprimée en contrôlant l'épaisseur du volume de mélange. Une bonne euristique est de choisir l'épaisseur proportionnelle au diamètre de la courbe d'intersection, défini comme la plus grande distance entre deux points de la courbe. En plus de permettre l'animation progressive d'un mélange en cas d'animation, ce choix de paramètre est intuitif durant les sessions de modélisation interactive, car les parties des objets qui s'interpénètrent le plus sont aussi celles qui auront le mélange le plus étendu. Si malgré tout l'utilisateur n'est pas satisfait par ce résultat il peut toujours ajuster manuellement l'épaisseur.

3.3 Union propre de surfaces implicites

Cette section décrit la manière dont nous combinons les fonctions de champ en dehors du volume de mélange pour éviter les mélanges indésirables. Pour cela nous utilisons un opérateur d'union implicite. Cependant, pour empêcher les artéfacts durant les mélanges ultérieurs cet opérateur doit être une "union propre" i.e. qui génère un champs C^1 continu dans l'espace, excepté sur le lieu de l'intersection. Il faut noter que l'opérateur max est le plus simple des opérateur d'union implicite mais il ne garanti malheureusement pas la continuité C^1. Comme nous l'avons déjà dit les R-fonctions de Rvashev [58] offrent une solution simple dans le cas des fonctions à support global, seulement une solution plus efficace que l'arc-d'une-élipse union proposée par Barthe [16] est nécessaire dans le cas des fonctions à support local. La figure 3.1 illustre l'importance d'un tel opérateur.

Dans cette partie nous montrons comment nous étendons les opérateurs de Rvashev aux surfaces implicites à support local, et de manière plus générale à tout type de surface implicite générée par un squelette, y compris les blobs à support global et les surfaces de convolution. De plus cette généralisation est effectuée avec un soucis d'efficacité d'évaluation. La figure 3.4 illustre les différentes opérations que nous appliquons aux fonctions de Rvashev pour obtenir nos nouveaux opérateurs.

3.3.1 Union propre à support local

L'opérateur d'union propre de Rvashev U_R, dont la partie positive est décrite dans la figure 3.4(b), a été élaboré pour combiner des fonctions similaires à des distances : isovaleur nulle, valeurs négatives à l'intérieur et positives à l'extérieur. La fonction de champs se comportant comme une distance ; elle n'est donc pas bornée. Cet opérateur simple et efficace est défini par :

$$U_R(f_1, f_2) = f_1 + f_2 + \sqrt{f_1^2 + f_2^2} \tag{3.2}$$

Notre premier objectif est d'appliquer cet opérateur à une fonction potentielle à support local dont l'iso-valeur est égale à $C > 0$ et dont le champ

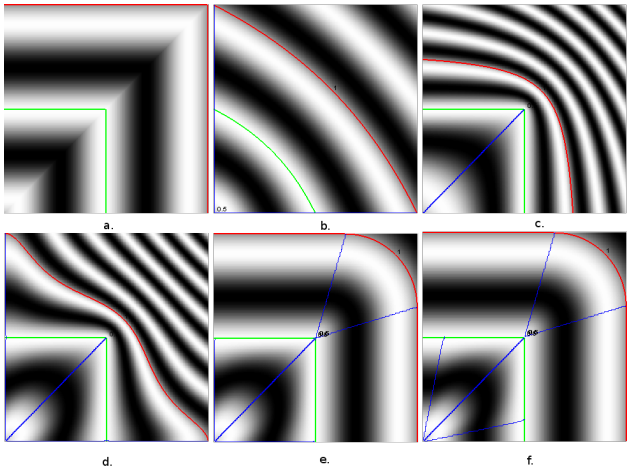

FIGURE 3.4 – Détail des étapes de construction de l'opérateur d'union propre : (a) union max, (b) opérateur de Rvashev brut, (c,d) étapes de construction de l'opérateur final (e-f).

scalaire décroit de l'intérieur vers l'extérieur de l'objet. Si les valeurs scalaires s'annulent à une distance donnée, alors le champ résultant doit lui aussi s'annuler à cet endroit. Cela signifie que l'opérateur doit être interne à l'ensemble des surfaces implicites à support local. Pour cela nous translatons et redimensionnons l'opérateur U_R ; nous obtenons l'opérateur U_{C_1} :

$$U_{C_1}(f_1, f_2) = C + U_R(f_1 - C, f_2 - C) \tag{3.3}$$

Puisque $U_{C_1}(0,0) = C(sqrt(2) - 1) \neq 0$, U_{C_1} doit être redimensionné pour satisfaire les conditions aux frontières. Cela donne :

$$U_{C_2}(f_1, f_2) = C(1 - (U_R(f_1 - C, f_2 - C)/U_R(-C, -C))) \tag{3.4}$$

Que l'on peut réécrire :

$$U_{C_2}(f_1, f_2) = \left(1 + \frac{\sqrt{2}}{2}\right) U_{C_1} - C\frac{\sqrt{2}}{2} \tag{3.5}$$

Le nouvel opérateur U_{C_2} (Voir figure 3.4c) respecte les propriétés essentielles des fonctions de champs à support local, mais malgré tout contracte le champs quand $f_1, f_2 > C$. En particulier en dehors des frontières de f_2, là où f_2 s'annule nous avons :

$$U_{C_2}(f_1, 0) = C + \frac{f_1 - C + \sqrt{(f_1 - C)^2 + C^2}}{1 + \sqrt{(2)}} \tag{3.6}$$

Par conséquent $U_{C_2}(f_1, 0) \neq f_1$, ce qui signifie que la fonction de champ f_1 est altérée même là ou f_2 n'a aucune influence sur f_1. Deux conséquences ressortent d'une telle propriété. Premièrement la qualité des champs souffre, après plusieurs mélanges, de compression alors même qu'aucun champs ne devrait les influencer. Deuxièmement il est courant en modélisation implicite d'utiliser des boites englobantes pour définir les régions qui ont besoin d'être mises à jours. L'absence de la propriété $U_{C_2}(f_1, 0) = f_1$ empêche cette optimisation cruciale.

Afin de préserver les propriétés de la fonction de champ initiale en dehors des zones d'influences réciproques nous interpolons entre un $max(f_1, f_2)$ standard à l'extérieur et notre fonction U_{C_2} à l'intérieur de la manière suivante :

$$U_{C_3}(f_1, f_2) = (1 - \delta(f_1, f_2))max(f_1, f_2) + \delta(f_1, f_2)U_{C_2}(f_1, f_2) \tag{3.7}$$

avec :

$$\delta(f_1, f_2) = \begin{cases} 0 & \text{quand } min(f_1, f_2) \leq 0 \\ \left(1 - \left(\frac{|f_1 - f_2|}{f_1 + f_2}\right)^2\right)^4 & \text{sinon.} \end{cases} \tag{3.8}$$

Comme illustré dans la Figure 3.4(d) le champ obtenu est lisse partout sauf au niveau de l'union des surfaces ($f_1 = f_2 = C$) et il se rapproche de f_1 (respectivement f_2) quand l'autre fonction s'annule.

Malheureusement la fonction potentielle est encore très contractée à l'intérieur de l'objet, là où les deux fonctions potentielles sont supérieures à l'iso-valeur $f_1 > C$ et $f_2 > C$. Au lieu de cela on voudrait une fonction de champs qui se comporte comme les opérateurs arc d'une ellipse [15]. Pour

cela on utilise l'opérateur de Barthe U_B [15] (Eq 2.8) à l'intérieur de l'objet, c'est à dire là où $f_1 > C$ ou $f_2 > C$, plutôt que U_{C_3}. La figure 3.4(e) montre que le champ scalaire généré devient alors lisse et ne comporte plus de compressions. Cet opérateur que nous appelons U_C, permet de garder un champ de bonne qualité au cours de mélanges répétés, permettant donc une utilisation répétée, par exemple au sein d'un modeleur implicite.

Équation finale de l'opérateur U_C :

$$U_C = \begin{cases} U_B(f_1 - C, f_2 - C) + C & \text{si } f_1 \geq C \text{ ou } f_2 \geq C \\ U_{C_3}(f_1, f_2) & \text{sinon} \end{cases} \tag{3.9}$$

Ce type de composition est très efficace mais provoque une discontinuité de type C^0 au niveau de l'iso-valeur de la surface implicite. Ce problème peut-être résolu facilement en interpolant à proximité de l'iso-valeur. Mais comme cette discontinuité se trouve au niveau de la surface elle se retrouve systématiquement à l'intérieur de l'objet lorsque l'on fait des mélange additif et ne provoque donc pas d'artéfact sur la surface implicite dans ce cas.

3.3.2 Adaptation aux surfaces de convolution à support global

L'opérateur U_C peut être appliqué aux surfaces de convolution à support local, mais pas de manière efficace aux surfaces de convolution à support global. Les surfaces de convolution à support global ont l'avantage d'être définies indépendamment d'une résolution donnée, elles simplifient la modélisation de formes comportant des détails d'échelles différentes. En contrepartie, ces surfaces n'ont pas un support fini, ce qui peut poser des problèmes de performance. En particulier l'influence d'une surface à support local s'arrête quand son champ potentiel s'annule. Ce cas n'arrive pas pour les fonctions à support global.

Si U_C était utilisé pour de telles fonctions, l'interpolation appliquée en dehors de l'objet ne serait jamais utilisée, le champs potentiel ne s'annulant nulle part. Notre choix est de modifier la fonction δ pour qu'elle conserve intacts les champs f_1 et f_2 sur un espace plus grand. Ainsi on peut éviter l'évaluation

couteuse de U_C dans de nombreux cas. En pratique on modifie la frontière d'interpolation $f_1 = 0$ en $f_1 = \alpha_1 f_2$ et réciproquement $f_2 = \alpha_2 f_1$:

$$\delta_c(f_1, f_2) = \begin{cases} 0 & \text{quand } f_1 \leq \alpha_1 f_2 \text{ ou } f_2 \leq \alpha_2 f_1 \\ \left(\frac{4(f_1 - \alpha_1 f_2)(f_2 - \alpha_2 f_1)}{(1-\alpha_2)f_1 + (1-\alpha_1)f_2}\right)^2 & \text{sinon.} \end{cases} \tag{3.10}$$

Une fois cela fait nous continuons à utiliser l'opérateur de Barthe U_B à l'intérieur du volume pour tirer profit des délimiteurs α_1, α_2, tout en conservant une fonction potentielle de qualité (voir Figure 3.4(f)).

3.4 Composition dans la zone de mélange

Dans cette section, nous discutons du choix de l'opérateur à l'intérieur même des volumes de mélange. Comme souligné dans l'introduction de ce chapitre, notre but est de choisir automatiquement les paramètres de mélange, pour éviter la dilution des petits détails quand ils sont mélangés à de gros objets. Notre opérateur doit aussi permettre une transition aussi douce que possible entre les régions de mélange, et les régions extérieures où notre nouvel opérateur d'Union propre U_C s'applique. Nous présentons la manière dont nous traitons ces deux problèmes dans les 2 sous-parties qui suivent.

3.4.1 Mélange qui préserve les petits détails

Nous avons vu dans l'état de l'art que des familles d'opérateurs avec de bonnes propriétés de localisation et de forme ont déjà été introduites par Barthe [15, 16] applicables aux fonctions de champs à support local et global. Plus précisément ces opérateurs restreignent la zone de mélange entre deux fonctions de champs de deux surfaces implicites. La forme du mélange est communément décrite par une courbe dessinée dans l'espace des fonctions f_1, f_2 (Figure 3.5 partie de gauche).

Pour empêcher les petits détails de se mélanger, la distance à laquelle le mélange doit s'effectuer sur le petit objet doit être relativement petite alors que la distance sur le grand objet peut être bien plus grande (Figure 3.5

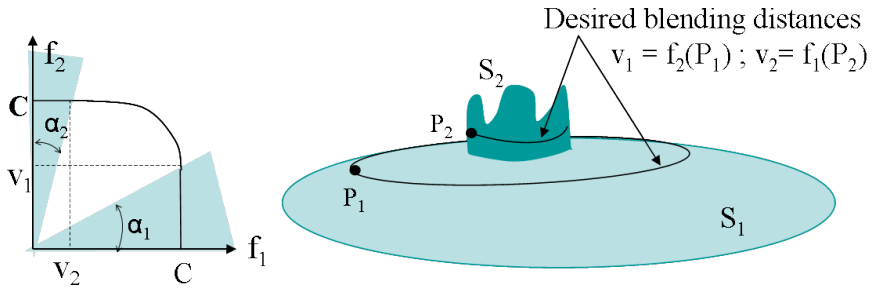

FIGURE 3.5 – A gauche : représentation de la fonction de mélange de Barthe, les zones bleues représentent les zones ou le potentiel est inchangé. A Droite : la limite de la zone de mélange est choisie à l'aide des points P_1 et P_2 qui servent à calculer respectivement les potentiels limites v_1 et v_2.

partie de droite). U_B nous laisse le choix des iso-valeurs v_1, v_2 délimitant la zone de mélange. Le choix de ces valeurs scalaires est simplifiée par la sélection judicieuse de points P_1 et P_2 sur les surfaces implicites S_1 et S_2.

Soit P_1 un point de S_1 qui se trouve à la frontière désirée pour le mélange. Alors P_1 vérifie $f_2(P_1) = \alpha_1 f_1(P_1)$. Or $f_1(P_1) = C$ car P_1 sur S_1. On en déduit que les valeurs à choisir pour α_1 et α_2 sont :

$$\alpha_1 = \frac{v_1}{C} \text{ avec } v_1 = f_2(P_1), \text{ ainsi que } \alpha_2 = \frac{v_2}{C} \text{ avec } v_2 = f_1(P_2) \qquad (3.11)$$

Le problème est maintenant de choisir automatiquement les distances séparant respectivement P_1 de S_2 et P_2 de S_1, telles que le mélange corresponde aux attentes de l'utilisateur. Cela permet d'éviter à l'utilisateur la manipulation directe des paramètres v_1 et v_2.

En pratique, nous choisissons une distance de mélange pour chaque surface qui est proportionnelle à la taille du détail le plus petit dont nous souhaitons empêcher la dilution, voir Figure 3.5. Parfois selon les types de surfaces implicites, les modèles contiennent déjà de bons estimateurs de la taille des détails. Par exemple, les représentations reposant sur des squelettes nous indiquent souvent les rayons de courbure minimaux des formes créées. Néanmoins ces informations peuvent ne pas être disponibles. On peut dans ce cas estimer la courbure de chacune des surfaces, soit explicitement à partir des

maillages des surfaces concernées, soit en utilisant directement les fonctions potentielles [39]. Il est intéressant de noter que l'on peut aussi utiliser le gradient du champs potentiel quand on utilise certaines primitives squelettes à rayon d'influence variable, ou alors des surfaces de convolution. Les petits détails ont alors un gradient plus important que les gros détails du fait de leur zone d'influence plus petite. Dans ces cas là, il est important d'avoir un opérateur de mélange bien défini, qui conserve cette propriété au cours de l'édition. En pratique dans MATISS nous avons conservé le même calcul de P_1 et P_2.

Quelle que soit la technique utilisée, nous calculons une estimation des rayons de courbure r_1 resp. r_2 des petits détails que nous voulons préserver sur les surfaces S_1 définie par f_1, resp. S_2 définie par f_2. Les distances de mélange d_1 sur S_1 et d_2 sur S_2 sont choisies comme proportionnelles aux rayons de courbures (nous utilisons en pratique : $d_1 = r_1/2$ resp. $d_2 = r_2/2$). Enfin pour calculer v_1 nous choisissons un point P_1 situé à une distance d_1 d'un point p de la courbe d'intersection dans la direction du gradient de f_2 en p. v_2 se calcule de manière analogue :

$$v_1 = f_2\left(p + d_1 \cdot \frac{\nabla f_2(p)}{\|\nabla f_2(p)\|}\right), v_2 = f_1\left(p + d_2 \cdot \frac{\nabla f_1(p)}{\|\nabla f_1(p)\|}\right) \tag{3.12}$$

Il peut être nécessaire de faire une moyenne des valeurs v_1, v_2 obtenue sur plusieurs points p quand le gradient du champ potentiel n'est pas suffisamment uniforme le long de la courbe d'intersection.

C'est ainsi que nous déterminons automatiquement les paramètres de l'opérateur U_B. Dans le cas où l'on ne souhaite pas le mélange progressif présenté dans la section 3.2.2, le rayon R de la section tubulaire qui entoure le mélange peut être déterminé, lui aussi, à l'aide de d_1 et d_2. Tout comme pour U_B nous souhaitons que la localisation du mélange soit de l'ordre de grandeur de la taille du plus petit détail. Dans la pratique nous utilisons donc $R = 2min(d_1, d_2)$.

3.4.2 Transition entre mélange et union

Notre but est maintenant de générer un opérateur U final, qui interpole, sans que ce soit visible, entre les zones de mélange, dans le volume de mélange, et

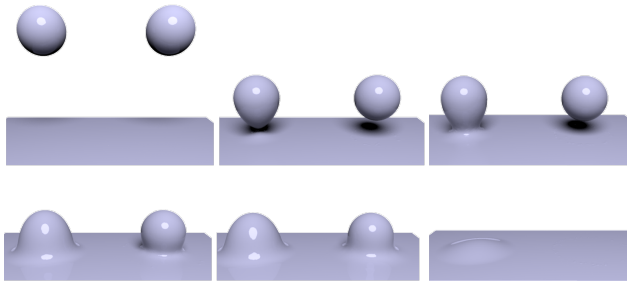

FIGURE 3.6 – Animation de mélange d'une goute et d'une flaque d'eau implicite. La goute de gauche utilise un opérateur traditionnel, se mélange à distance et est déjà complètement mélangée lorsque le mélange local commence de l'autre coté.

les zones d'union en dehors. Pour que la transition soit invisible il faut que le champs soit C^1 sur les bords du volume de mélange. L'opérateur U est le résultat de l'interpolation entre U_B et U_C :

$$U(f_1, f_2) = (1 - \beta)U_C(f_1, f_2) + \beta U_B(f_1, f_2) \tag{3.13}$$

avec β une puissance de la valeur v de la fonction potentielle du volume de mélange.

3.5 Résultats

Pour démontrer la pertinence de notre méthode pour l'animation, nous l'illustrons par le mélange progressif d'une goutte dans un volume de liquide. Nous avons également implémenté notre technique dans MATISS (Chapitre 2), ce qui nous permet d'illustrer également notre nouveau mélange dans une application de modélisation intuitive.

La Figure 3.6 montre l'animation simple de goutte d'eau qui tombe sur une flaque d'eau. Une primitive locale (soft object) est utilisée pour la goutte alors qu'un champs potentiel type distance est utilisé pour modéliser le plan. Sur la figure, la goutte de gauche utilise un mélange standard type somme alors que celle de droite utilise notre technique. Ce petit exemple valide notre choix de ne commencer le mélange que lorsque les objets entrent en contact

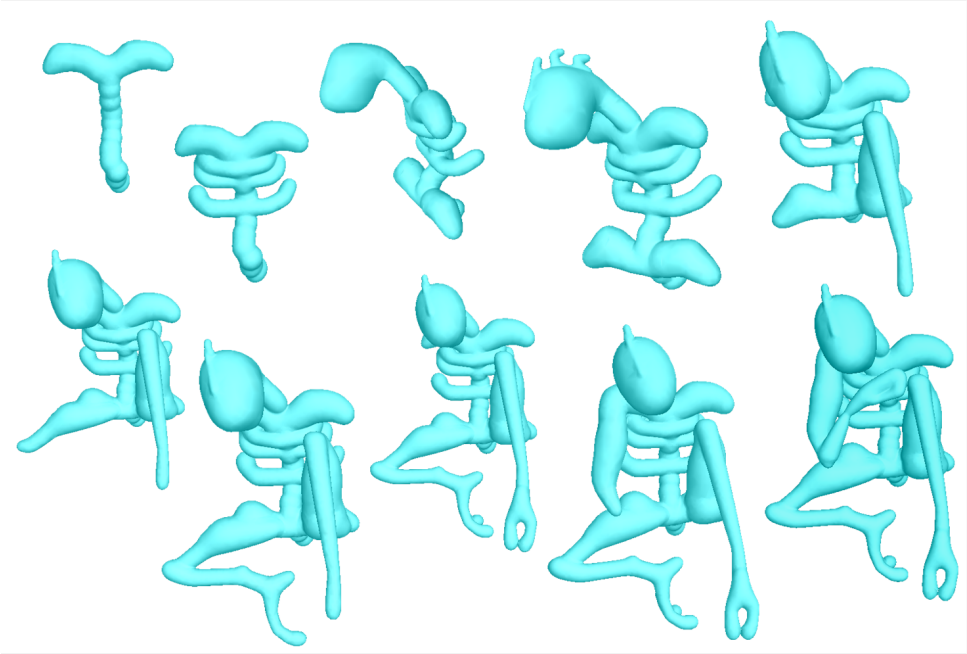

FIGURE 3.7 – Détails de certaines étapes de la création progressive d'un personnage d'Alien penseur, à l'aide du système MATISS et du nouveau mélange local.

et donc s'intersectent. Cela valide aussi l'euristique que nous avons choisie pour calculer la taille de la zone de mélange.

Bien sur, nous avons aussi utilisé ce nouvel opérateur dans MATISS puisqu'il a été initialement pensé pour. La Figure 3.7 montre les étapes de création d'un penseur extraterrestre. Le penseur et les danseurs (Figure 3.9) illustrent les propriétés de mélange local de notre opérateur. Le dragon et l'arbre (Figure 3.8) d'un autre coté illustrent la préservation des petits détails quand ils sont fusionnés avec des gros objets, en l'occurrence le relief sur le ventre du dragon et les morceaux d'écorce de l'arbre.

3.5.1 Temps de calcul

L'artiste qui a créé ces formes à passé de 30 minutes à une heure sur chacune d'elles. Ce temps inclue tous les temps de calculs y compris les annulations.

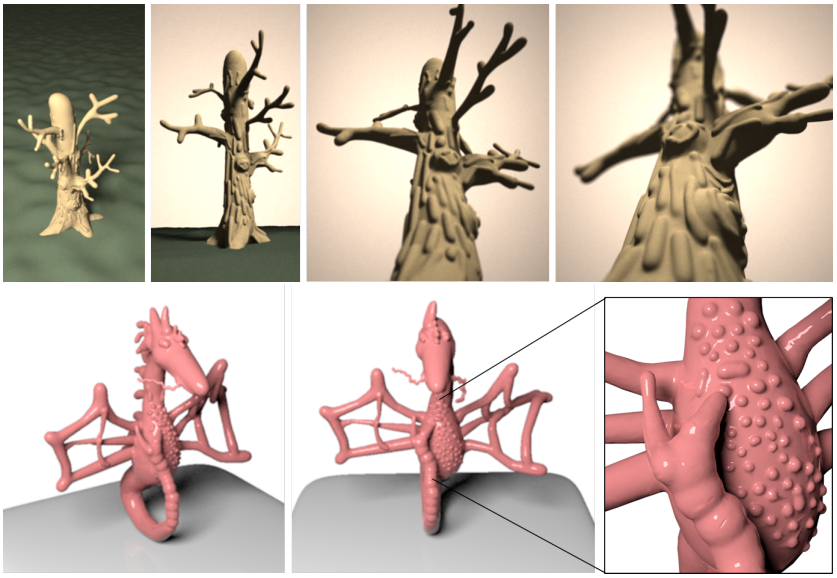

FIGURE 3.8 – Deux exemples aboutis créés avec MATISS puis dessinés à l'aide d'un logiciel de rendu.

Pour donner une meilleure idée du temps nécessaire à la création des modèles à partir de zéro nous avons enregistré l'historique de construction pour chaque exemple et nous avons minuté la proportion de temps passée à calculer le mélange durant le chargement des objets. Ce temps peut alors être divisé par le nombre de primitives pour se faire une idée du temps moyen nécessaire pour un mélange. Vous pouvez trouver ces résultat dans le tableau 3.1.

Nous comparons aussi dans le Tableau 3.2 le cout d'évaluation de différents opérateurs de mélange. Comme prévu plus l'opérateur est sophistiqué plus le temps de calcul est important. Néanmoins l'avantage des opérateurs tels que U_B et U_C est leur localité. Par conséquent dans des modèles de grande taille ils nécessitent moins d'évaluations même si leur évaluation est plus couteuse.

3.5.2 Mailleur Optimisé

L'opérateur que nous venons de présenter nous permet d'effectuer une optimisation significative de l'étape de maillage de notre surface. D'une manière générale notre opérateur nous permet d'augmenter la cohérence de la surface

Modèle	Dragon	La ronde	Le penseur	L'arbre
Nombre de primitives	68	23	24	54
Temps de calcul brut de reconstruction	277	34	41.9	475
Temps de calcul moyen par primitive	2.65	1.24	1.31	6.3
Temps de calcul moyen par mélange	1.43	0.25	0.43	2.48

TABLE 3.1 – Temps de calculs (en secondes) nécessaires pour générer les modèles qui illustrent ce chapitre. Le dragon et l'arbre, bien que constitués de nombreuses primitives ne souffrent pas de temps de calcul trop important. La version de l'arbre utilisé pour ce test contient environs 100000 sommets tandis que la version finale en contient 400000.

U_R	U_B	U_C
0.08	0.78	0.95

TABLE 3.2 – Temps de calcul en secondes pour un million d'évaluation de chacun des différents opérateurs de composition utilisés dans la Figure 3.4

implicite, car on sait maintenant plus précisément quand et où surviennent les changements topologiques. Ils surviennent uniquement quand deux surfaces s'intersectent et dans un rayon donné autour du contour d'intersection. Cette cohérence est importante car elle permet d'utiliser le maillage précédent comme base pour construire le nouveau maillage. La technique est similaire à celle utilisée dans MATISS excepté qu'il suffit de remailler la partie du maillage se situant à l'intérieur du volume de mélange. Il n'est plus besoin de propager le mélange à partir du contour d'intersection tant que l'influence réciproque des volumes à mélanger n'est pas négligeable.

3.5.3 Discussion

Grâce à notre nouvelle méthode, MATISS et en particulier la conservation de la netteté des petits détails, a été jugé plus intuitif à utiliser qu'avec l'ancien mélange hybride. Dans MATISS les choix des paramètres de mélange sont restés invisibles à l'utilisateur. Ce faisant nous avons conservé la simplicité d'utilisation de notre logiciel tout en produisant un résultat plus conforme aux attentes de l'utilisateur. Cela nous a aussi permis d'améliorer grandement la stabilité du logiciel, en particulier du mailleur, qui éprouvait des difficultés

FIGURE 3.9 – Création faite dans MATISS, inspirée de la peinture des danseurs de l'artiste Matisse.

à mailler des formes qui s'intersectent mais restent constamment proches par ailleurs.

Dans un cadre plus général, des solutions à trois difficultés majeures du mélange de surfaces implicites, sont apportées par notre méthodes : le mélange imprévisible, le manque de localité et la dilution des petits détails. Les bonnes propriétés de continuité de notre fonction de champs et l'absence de contraction du champ après chaque mélange permet la création de modèles complexes, par assemblages successifs de primitives.

Ce travail nous a aussi permis d'introduire un nouvel opérateur d'union propre qui peut-être utilisé tel quel pour combiner des primitives quand on souhaite une union booléenne des formes. Cet opérateur était pensé à l'ori-

gine pour pouvoir effectuer une transition continue entre un mélange lisse et une union booléenne. C'est en quelque sorte un travail inachevé dont l'idée fondatrice a été reprise pour créer l'opérateur utilisé dans le gradient blending [6].

Enfin nous avons aussi identifié une liste des faiblesses de notre méthode que nous exposons ici :

Premièrement, dans notre implémentation actuelle, le volume de mélange est défini à l'aide d'une primitive à support local, de type Soft-Object. Comme cette dernière est calculée en utilisant la distance à la courbe d'intersection, le champs généré peut, selon l'épaisseur du volume de mélange, ne pas être C^1 dans les parties concaves de la courbe. En pratique nous n'observons pas d'artéfacts car le rayon du volume de mélange est souvent plus petit que le rayon de courbure, et la plupart des concavités de la courbe se trouvent à l'intérieur de l'objet. Vous pouvez cependant observer cet artéfact sur la Figure 3.3 en bas à droite, là ou la fonction de champ semble étrange au milieu de la zone d'intersection entre la flaque et la goute. Pour résoudre cela, on pourrait utiliser des primitives de convolution à support compact pour définir la zone de mélange. Cela garantirait une continuité C^1 de la fonction de champs tout aillant un comportement similaire, ainsi qu'un surcout raisonnable.

La deuxième faiblesse que nous voyons à notre méthode est que le calcul automatique des paramètres de mélange n'est certainement pas aussi général que nous le souhaiterions. En particulier deux objets que nous mélangeons peuvent déjà contenir un lot de petits détails à leur surface alors que nous souhaitons les mélanger à une grande échelle. Analyser dans le domaine fréquentiel les différentes échelles de détails et laisser l'utilisateur choisir parmi ces échelles, lors de chaque mélange, serait la solution idéale. En pratique MATISS souffre peu de ce problème car la modélisation se fait de manière progressive et par raffinement successifs ; les petits détails sont généralement ajoutés en dernier.

Finalement bien que MATISS fonctionne de manière interactive, le remaillage local prend une à deux secondes. Le système bénéficierait grandement d'un temps plus court d'une part, mais encore plus d'une manipulation en temps réel des primitives implicites. Pour cela nous aurions besoin d'un meilleur mailleur ainsi que d'un cache efficace pour stocker les potentiels. Une solution idéale se situerait peut-être entre le mécanisme de cache de potentiel proposé par Schmidt [66] et le "Twin Mesh" de Bouthors et Nesme [26].

3.6 Conclusion

Dans ce chapitre nous avons présenté une méthode originale, qui ouvre de nouvelles perspectives pour le mélange de surfaces implicites. Notre méthode résout plusieurs limitations majeures de la modélisation fonctionnelle de surfaces implicites. Elle peut être entièrement automatisée, ou à défaut, quelques contrôles simples peuvent être laissés à l'utilisateur. Au lieu de laisser les surfaces se mélanger inexplicablement, nous proposons une méthode ou le mélange est prévisible et local, tout en conservant la force des surfaces implicites : leur capacité à se mélanger de manière lisse.

La plupart des opérateurs de mélange sont binaires alors que la somme peut-être n-aire. Concrètement un opérateur n-aire peut-être intéressant s'il permet d'éviter des évaluations des champs potentiels négligeables.

Deuxième partie

Modélisation Interactive de Surfaces : Application au design de terrains

Cahier des charges pour une modélisation interactive intuitive

Dans la première partie nous avons présenté un modeleur volumique, que nous avons souhaité aussi intuitif que possible. Pour cela nous avons introduit une métaphore de peinture et nous avons utilisé un modèle basé sur des surfaces implicites afin de générer la géométrie. Dans cette seconde partie nous nous intéressons à la modélisation progressive d'une surface. Dans les chapitres 4 et 5, nous nous concentrons sur la modélisation de terrains, qui constituent un type de surface particulier. Nous présentons aussi dans la conclusion du manuscrit un prototype qui tente d'étendre notre approche à la modélisation de surfaces libres quelconques. Ces travaux utilisent un socle technique commun, composé d'un solveur multi-grille et de pavage (tesselation en Anglais) sur GPU (décrit en détail dans le chapitre 4). Pour choisir ce socle technique nous nous sommes appuyés sur l'expérience utilisateur que nous souhaitions offrir dans ces applications. Ce préambule à la seconde partie de cette thèse détaille l'expérience utilisateur que nous souhaitions créer : d'abord de manière générale dans la première section, pour développer un peu plus certains de ces points dans les sections suivantes.

Motivations

Après avoir terminé les travaux sur MATISS, nous souhaitions nous intéresser à la modélisation de terrains par croquis, vue de la première personne, c'est à dire permettant à l'utilisateur de redessiner certains contours depuis son point de vue courant, lors de l'exploration d'un terrain. Nous souhaitions

que l'utilisateur puisse modéliser comme s'il croquait, d'où il se trouve, un paysage qu'il contemple. Ce travail est présenté dans le chapitre 5.

La littérature comporte un nombre important de travaux sur la génération de terrains (voir 1.1.2). Tous ces travaux utilisent des méthodes différentes de génération ; rien de surprenant jusqu'ici. Seulement chacune de ces techniques implique un choix fort en terme d'expérience utilisateur. Certaines techniques sont de type vectorielles, génératives, et d'autres sont plus proches de Paint®, composées de petites éditions successives, mais en général obligent l'utilisateur à faire son édition en vue de dessus, comme s'il dessinait une carte. Le temps de calcul diffère bien sûr beaucoup d'une technique à l'autre, tout comme le type et le degré de contrôle donné à l'utilisateur. Ces caractéristiques techniques ont un impact fort sur l'expérience utilisateur ; il était donc important de définir avant tout, les grandes lignes de l'expérience utilisateur que nous souhaitions offrir. En particulier, le niveau d'interactivité et le niveau de contrôle.

Définition de l'expérience utilisateur souhaitée

Commençons par citer, telle qu'énoncée dans la norme ISO 9241-210, la définition d'expérience utilisateur : Ce sont "Les perceptions et les réponses, d'un sujet, résultant de l'utilisation ou de l'anticipation de l'utilisation d'un dispositif, d'un système ou d'un service". Cette définition ne s'intéresse qu'indirectement aux fonctionnalités d'un système, elle s'intéresse à l'effet que le système produit sur l'utilisateur. L'effet que nous recherchons est l'adéquation du modeleur à la création numérique. Nous cherchons à maximiser le pouvoir créatif au sein du modeleur et nous pensons que cela passe par :

- Un environnement de création cohérent, d'usage instinctif, qui évite la multiplication des outils complexes.
- Un environnement adapté à un usage prolongé
- Qui fonctionne comme un support de création. Qui autorise les essais, erreurs, les corrections.

Pour créer des environnements modernes propices à la création, quelques aspects nous tenaient particulièrement à cœur :

- Un modeleur rapide : Modéliser est comme un dialogue entre le créateur et son œuvre ; dialogue qui ne doit être coupé, au risque de perdre le fil

de la conversation. Pour cette raison, le temps entre deux actions d'édition, pendant lequel l'application est bloquée, ne peut dépasser quelques secondes sans altérer le processus créatif.
- Prévisible, contrôlable : Permettre autant que possible à l'utilisateur d'éditer l'objet comme il le conçoit mentalement, comme il se le représente. Pour minimiser les essais / erreurs, l'utilisateur doit pouvoir prédire facilement le résultat de l'action qu'il entreprend. Les algorithmes fractals sont souvent peu prévisibles.
- Liberté d'exécution : Nous souhaitons éviter les processus rigides, qui forcent l'utilisateur à effectuer les opérations dans un ordre prédéfini ou qui l'empêche de retoucher une modification passée.
- Échelle de modification : Les objets traités par les ordinateurs actuels sont complexes, comportent de petits détails. Nous voulions un modeleur capable de supporter le passage à l'échelle.

Voici notre cahier des charges. Certains de ces points méritent approfondissement, ce qui est fait dans les sections suivantes.

La modélisation temps réel pour un couplage Sensori-Moteur

Notre avis sur l'intérêt de la rapidité d'exécution a beaucoup évolué au cours de la thèse. Le travail sur les terrains nous a placé devant un dilemme : nous voulions gagner soit en vitesse soit en contrôlabilité par rapport à l'existant. Pour cela nous nous sommes basés sur la méthode de Hnaidi [42], déjà très rapide. Grâce à des optimisations complémentaires, la méthode est devenue suffisamment rapide pour envisager un changement important de technique d'interaction :

Plutôt que d'attendre que l'utilisateur finisse de dessiner pour générer le modèle, nous pouvions générer le terrain quasiment à la même fréquence que le taux de rafraichissement de la scène. On pouvait alors envisager de voir les reliefs d'un terrain se former en même temps que l'on dessine une silhouette de montagne ou que l'on saisit un point particulier du terrain pour le surélever. Cette idée s'est imposée au cours d'un séjour au Brésil, pour collaborer avec l'équipe VisGraf du professeur Velho, à l'IMPA de Rio de Janeiro.

En apparence simple, ce changement n'est pas anodin : il introduit une rupture en terme d'interaction et d'expérience utilisateur. L'utilisateur peut avoir un retour visuel permanent sur l'objet qu'il déforme ou manipule. Il peut voir à chaque instant quel effet produit le geste qu'il effectue, le paramètre qu'il modifie. Il peut donc ajuster son geste en temps réel. Une analogie avec la robotique permettrait de dire que le système permet de développer un couplage sensori-moteur.

Le couplage sensori-moteur, titre de cette section, désigne la création d'une relation entre un système sensitif et un système moteur. Le système sensitif permet d'évaluer les conséquences d'une action moteur, qui permet d'ajuster le système moteur en retour. On parle de couplage car cette relation est une boucle qui crée un lien étroit entre senseur et moteur. Dans le cas d'un modeleur, le couplage se base sur une boucle visualisation/action/visualisation/... Pour que l'effet fonctionne il faut que cette boucle tourne en temps réel ; de façon à ce que l'utilisateur ait un retour visuel immédiat, pouvant entrainer une correction. Par analogie, l'utilisation d'un retour haptique vise aussi à établir un couplage sensori-moteur, important pour les applications de chirurgie.

Dans le cadre du couplage visuel nous pouvons considérer deux fréquences importantes. Une fréquence entre une image par seconde et une image toute les trois secondes permet un couplage minimal, qui induit une association, par l'utilisateur, entre l'action et le résultat, mais ne permet pas une immersion. Une fréquence supérieure à 10 images par seconde augmente significativement l'immersion. L'utilisateur peut alors oublier encore plus l'outil pour se concentrer sur son modèle.

C'est pour créer ce second type de couplage sensori-moteur que nous souhaitons que notre modèle puisse être mis à jour plusieurs fois par seconde.

Favoriser la liberté d'exécution

Chacun pense la création d'une manière différente. Pour dessiner une même scène, certains préfèreront se concentrer d'abord sur un détail ; d'autres adopteront une approche différente, plus globale. C'est ce que nous appelons liberté d'exécution, la possibilité de choisir l'ordre dans lequel effectuer chaque

modification, la possibilité de retoucher une modification antérieure.

Dans MATISS il n'est, par exemple, pas facile de modifier/supprimer une primitive qui a déjà subit des mélanges ultérieurs. Dans l'implémentation actuelle il faut annuler les mélanges ultérieurs, effectuer la modification et enfin ré-appliquer les mélanges. C'est pour nous l'une des principales lacunes du système.

Nous voyons deux manières de garantir la liberté d'exécution. La première est de fournir des outils avec lesquels il n'est pas besoin de revenir en arrière. On peut dire que le système de calques de Photoshop^{TM}répond à ce besoin. C'est cette couche intermédiaire, qui nécessite une implémentation spécifique, qui permet la liberté d'exécution ; pas les outils de "<peinture">, dont l'application n'est pas commutative.

La seconde manière de garantir la liberté d'exécution est d'utiliser ce que nous appelons des méthodes génératives ou procédurales. Les méthodes procédurales sont génératives car un nombre réduit de paramètres permet de regénérer un résultat. Ainsi l'ordre des modifications importe peu car à chaque modification de paramètre, il "<suffit"> de tout regénérer. La génération procédurale doit donc être très rapide, pour pouvoir s'effectuer en temps réel.

Les représentations vectorielles, pour peu qu'elles soient assez rapides, constituent une famille de méthodes procédurales qui sont tout à fait adaptées à ce que nous souhaitons faire. C'est l'approche que nous avons choisi d'utiliser dans la suite de notre travail.

Les deux manières de garantir la liberté d'exécution ont leurs forces et faiblesses. Bien qu'elle soit plus couteuse en temps de calcul, une représentation vectorielle permet à l'utilisateur de modifier des caractéristiques (features) d'un terrain dans l'ordre qu'il souhaite, pouvant affecter globalement le terrain. Cette caractéristique favorise l'ajustement, la modification et l'édition pendant une longue période de temps.

Bilan

Si ces considérations ont pu évoluer un peu au cours de la thèse, elles ont toujours guidé l'élaboration des modeleurs de cette seconde partie. Le temps réel, le contrôle et la capacité à supporter des modèles de grande taille sont les points qui nous ont permit d'approcher l'expérience utilisateur que nous souhaitions.

Chapitre 4

Modeleur interactif de terrains

4.1 Introduction

Dans ce chapitre nous nous intéressons à la modélisation de terrains. Ce domaine a déjà été exploré largement comme nous avons pu le voir dans l'état de l'art (section 1.1.2). Ce que nous proposons ici est un système 3D de sculpture interactive de terrains qui permet de générer un terrain de grande taille en temps réel, à plus de 15 images par secondes. Le terrain est modelé à l'aide de primitives vectorielles dont la hauteur, la normale, la rugosité et le degré de continuité peuvent être choisi. Comme le travail de Hnaidi [42], nous utilisons une approche multi-grille pour générer le terrain, mais formulée différemment, basée sur l'équation bi-harmonique au lieu de l'équation harmonique, ce qui permet un contrôle plus intuitif des gradients. Grâce à une méthode de rendu optimisée nous sommes capables d'afficher en temps réel ce terrain pour un cout minimal.

Ces caractéristiques font que l'utilisateur peut voir le terrain se déformer en même temps qu'il manipule et modifie les primitives. Le résultat est un modeleur qui s'apparente à de la sculpture, entièrement en 3D, temps réel et immersif.

Enfin le système permet l'édition de terrains existants, donnés sous la forme de carte d'élévation (DEM). La formulation bi-harmonique rend la vectorisation aisée ; et nous proposons, de plus, une méthode pour joindre de manière

lisse le terrain généré et le terrain existant.

Par opposition au modeleur présenté dans le chapitre suivant, qui est dédié aux croquis de montagnes, nous qualifions ce premier modeleur de généraliste. Il n'est pas dévolu à un type d'édition en particulier, et c'est d'ailleurs ce modeleur qui est réutilisé et combiné à une interface spécifique dans le chapitre suivant.

4.1.1 Vue d'ensemble

Il a été important de travailler trois points en particulier pour faire ce prototype. La première difficulté était de créer une représentation de terrain qui supporte les fonctionnalités d'édition que nous souhaitions pour les terrains : contrôle fin de l'élévation du terrain, de son gradient, avec un nombre minimum de primitives vectorielles. Pour cela nous avons utilisé l'équation bi-harmonique dans un solveur multi-grille, ce qui nous a permis d'offrir les contrôles souhaités ainsi que d'excellentes performances. La seconde difficulté était de supporter l'édition de terrains de grande taille, en 3D temps réel. Pour cela nous avons déporté un maximum de calculs sur GPU (rendu, pavage et solveur), le seul calcul restant sur CPU étant la mise à jour d'un quad-tree adaptatif. Le troisième point était de développer une interface pour l'édition 3D temps réel de terrains.

Pour cette raison ce chapitre est structuré en quatre grandes parties :
- En tout premier nous présentons globalement le modeleur réalisé, pour donner au lecteur une vision d'ensemble du travail réalisé.
- Dans la seconde partie nous présentons le solveur multi-grille que nous utilisons. Nous présentons les fonctionnalités offertes par le solveur puis la manière dont nous avons implémenté chaque fonctionnalité à l'aide de ce dernier : l'édition vectorielle, le contrôle des normales et de la forme, le recollement avec un terrain existant, la vectorisation.
- L'efficacité du solver GPU permet de générer beaucoup de données, qu'il faut pouvoir afficher de manière efficace. Pour afficher en temps réel la géométrie du terrain générée par le solveur nous combinons un quad-tree adaptatif avec du pavage sur GPU (tesselation), ce qui permet de réduire la bande passante CPU-GPU. C'est ce que nous présentons dans la troisième partie de ce chapitre.

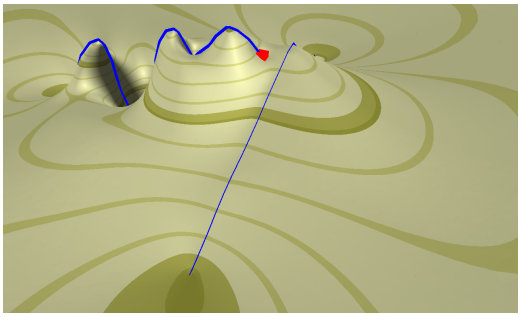

FIGURE 4.1 – : Les objets bleus et rouge apparaissant à la surface du terrain sont des primitives vectorielles qu'il est possible de manipuler à l'aide des ancres (cubes rouges) que l'on peut observer.

– Enfin nous présentons l'interface que nous avons élaborée, qui permet l'édition intuitive et temps-réel de terrains de grandes tailles. Cela est possible grâce à la rapidité et au contrôle fournis par les algorithmes que nous avons choisi.

4.2 Chaine de traitement du logiciel d'édition interactive

L'idée centrale de notre modeleur est que l'utilisateur doit pouvoir éditer le terrain en temps réel. Pour cela nous lui donnons des Primitives 3D à manipuler (Figure 4.1). Ces primitives permettent d'éditer le terrain de manière vectorielle ; elles sont simples dans ce chapitre (point, courbe), mais peuvent être plus élaborées comme dans le chapitre suivant. Quand l'utilisateur déplace une primitive, le terrain se déforme en temps réel pour suivre l'action de l'utilisateur. La manière dont l'édition, les contraintes, la génération et le rendu de la géométrie s'articulent est décrite dans cette partie. Voici ces éléments présentés un à un.

Les primitives : Lors de l'édition l'utilisateur contrôle des primitives 3D dont l'interface est pensée pour être aussi intuitive que possible. C'est en déplaçant, en déformant, en modifiant les propriétés de ces primitives que l'utilisateur sculpte le paysage.

Les contraintes : Les Primitives sont l'interface utilisateur visant à faciliter la manipulation du terrain. Elles ne sont pas utilisées telles qu'elles lors de la génération du terrain. La génération du terrain, quant à elle, passe par un solveur qui prend en entrée des contraintes : contraintes d'altitude, de gradient, de bruit, de recollement, etc. Les primitives doivent donc être converties en contraintes pour être utilisées par l'étape de génération.

Le solveur : Le solveur est la partie chargée de convertir les contraintes en une carte d'élévation, de bruit, etc. C'est aussi l'étape la plus consommatrice en calcul, c'est pour cela qu'elle s'effectue sur le GPU.

La conversion des primitives en contraintes : Dans notre cas les contraintes sont passées au solveur sous la forme d'une texture de contraintes. Pour créer cette texture les primitives sont converties en entités géométriques OpenGL qui sont dessinées dans la texture de contraintes. Pour dessiner les contraintes nous utilisons des programmes sur GPU qui sont spécifiques au dessin de contraintes ; cela nous permet d'encoder la texture de contraintes dans un format lisible par le solveur.

Génération de la texture de résultat La texture de contraintes est alors passée au solveur qui génèrera la texture résultat. C'est à ce moment que l'algorithme multi-grille intervient : les contraintes sont sous-échantillonnées jusqu'à la plus petite échelle de grille puis le processus de résolution/sur-échantillonnage commence. Une fois l'algorithme terminé, la texture résultat a la même taille que la texture de contraintes initiale.

Rendu de la texture résultat : La texture de résultat est particulièrement volumineuse. Pour pouvoir mettre à jour la géométrie en temps réel, à une fréquence de 20 fps, nous devons faire attention à l'algorithme de rendu que nous utilisons. En particulier il est bon d'éviter les transferts CPU/GPU inutiles ainsi que l'affichage d'une géométrie trop détaillées. Avec de petits volumes de données les calculs peuvent être effectués sur le CPU. Le nouveau modèle peut être mis à jour et transféré au GPU pour être affiché. Le temps de transfert CPU/GPU devient rapidement limitant quand la taille des modèles augmente et quand la fréquence de mise à jour augmente.

Structure en Quad-Tree avec pavage : Pour afficher le terrain nous utilisons un Quad-Tree adaptatif grossier qui est généré sur CPU. Ce Quad-Tree est alors envoyé sur GPU, subdivisé sur GPU à l'aide des récentes unités de tesselation présentes sur GPU. La mise à jour du quad-tree est guidée par le position de la caméra ainsi qu'un sous-échantillon du terrain, rapatrié depuis le GPU.

Dans les sections suivantes nous exposons avec plus de détails ce que nous avons présenté ici grossièrement.

4.3 Solveur Multi-grille

Maintenant que nous avons présenté la vue d'ensemble du modeleur intéressons nous à la pièce maitresse du système : le solveur. C'est la partie chargée de transformer les contraintes en terrain. Comme dans le travail de Hnaidi notre solveur génère une carte d'élévation, une carte contenant les paramètres de bruit procédural ainsi qu'un masque utilisé pour les DEM. C'est l'étape la plus consommatrice en temps de calculs.

Le solveur que nous présentons permet d'exercer des contraintes de position, de normale et aussi de contrôler localement le degré de lissage des arrêtes.

Dans la première sous-partie nous présentons ce qui sont pour nous les principaux avantages et inconvénients des solveurs multi-grille. Ensuite nous présentons comment à l'aide d'une formulation bi-harmonique nous arrivons à contrôler élévations et gradients. Nous montrons comment l'utilisateur peut contrôler la forme du terrain en contrôlant localement le type d'équation résolue diffusion ou bi-harmonique. Enfin nous montrons comment nous implémentons l'édition de terrains réels.

4.3.1 Choix d'un solveur multi-grille

Nous utilisons dans tout ce travail un solveur multi-grille que nous adaptons à nos besoins à de multiples reprises et qui nous permet d'obtenir de très bonnes performances de modélisation ; c'est pourquoi cela vaut la peine d'expliquer quels sont les avantages et les inconvénients de tels solveurs. En voici quelques aspects :

Optimalité : Parmi les avantages on peut noter que la résolution des certaines équation est optimale à l'aide de tels solveurs. Par chance c'est le cas de l'équation harmonique et de l'équation bi-harmonique que nous utilisons ici.

Fait pour le GPU : La résolution multi-grille est particulièrement aisée lorsque le calcul s'effectue sur une grille régulière ; grille qui est une texture sur GPU. Les grilles de résolution inférieures correspondent aux niveaux de Mip-Map de la texture. Pour chaque itération du solveur, chaque sous-échantillonnage, chaque sur-échantillonnage, le traitement des cellules (pixels) de la grille (texture) s'effectue de manière indépendante. Les algorithmes multi-grille peuvent aussi fonctionner sur des grilles non-régulières ; on parle alors de multi-grille algébrique. Ce type de multi-grille est un peu moins parallélisable, mais garde toutefois de très bonnes propriétés.

Adaptabilité : Nous avons observé lors de ce travail qu'il est assez aisé de modifier localement les équations résolues par le système. Si bien qu'il est possible de résoudre certaines équations non-linéaires sans plus de difficultés que la résolution d'une équation harmonique. Cela n'est bien-sur pas toujours le cas et certaines équations convergent tout simplement trop lentement.

Adaptatif : De part sa nature multi-échelle, le multi-grille s'adapte facilement à la résolution sur un quad-tree. Nous n'avons toutefois pas implémenté cette approche, Ce qui nous aurait permis d'optimiser significativement notre algorithme de génération de terrain.

Sensibilité à la formulation du problème : La complexité des solveurs multi-grille vient du fait qu'ils peuvent être sensibles à la formulation d'un problème. Un même problème, formulé de deux manières différentes peut avoir des taux de convergence drastiquement différents. C'est par exemple le cas de l'équation bi-harmonique, comme nous le verrons dans la section suivante.

Sensibilité aux schémas de changement d'échelle : Un autre problème délicat est l'échantillonnage, aussi bien le sous-échantillonnage des con-

traintes que le sur-échantillonnage des résultats temporaires. Une équation peut gagner ou perdre un ordre de grandeur de convergence selon l'échantillonnage utilisé.

4.3.2 Équation bi-harmonique : Motivations

Dans ce travail nous utilisons l'équation bi-harmonique pour prendre en compte les contraintes de gradient. Nous expliquons ici pourquoi.

Pour générer le terrain nous sommes inspiré de la méthode utilisée par Hnaidi [42]. Hnaidi résout une équation de diffusion à l'aide d'un solveur multi-grille. La méthode s'inspire du travail réalisé par Orzan sur les "diffusion curves" [56] et permet de générer quasi instantanément un terrain en interpolant des contraintes vectorielles. Ces deux travaux reposent sur la résolution de l'équation de diffusion, ou équation harmonique (voir section 4.3.3), qui tend à annuler en tout point le Laplacien, c'est à dire la variation du gradient. Pour se faire une idée du comportement de l'équation harmonique, on peut remarquer qu'en 1D, résoudre un système de contraintes correspond à interpoler linéairement ces contraintes. La surface obtenue est C^0. De fait l'équation harmonique engendre des discontinuités de gradient autour des contraintes, et permet seulement d'interpoler des contraintes de position, non des contraintes de gradient. Pour effacer les discontinuités autour des contraintes, Orzan applique un lissage à proximité des contraintes dont le rayon d'influence est fixé par l'utilisateur. Utiliser une telle approche pour la modélisation de terrain n'offrirait pas un grand contrôle, car l'utilisateur ne peut alors pas fixer la valeur du gradient.

Pour permettre un contrôle des gradients Hnaidi [42] utilise une autre approche qui consiste à scinder la résolution des gradients et des altitudes en deux problèmes de diffusion séparés. Il commence par générer une carte de gradient en interpolant, grâce à la diffusion, des contraintes de gradient. Ensuite il utilise cette carte de gradient pour guider la diffusion et ainsi prendre en compte des gradients pré-calculés. Cette formulation permet de guider la diffusion mais ne permet pas de faire en sorte que la normale de la surface soit localement celle voulue par l'utilisateur. L'équation sous-jacente reste C^0, ne permet pas d'interpoler les gradients, ni de leur fixer une valeur.

Pour cette raison nous avons cherché une formulation mathématique per-

mettant d'interpoler et de fixer les gradients. Cette recherche nous a amené à formuler la génération de terrain comme la résolution d'une équation bi-harmonique, garantissant ainsi la continuité C^1 de la surface ainsi générée (Voir Figure 4.2). La formulation commune des système bi-harmonique permet de contraindre facilement la courbure, non le gradient ; nous avons donc trouvé une formulation de l'opérateur bi-harmonique permettant d'inclure un contrôle aisé et intuitif du gradient.

Avoir une formulation bi-harmonique nous offre de nombreux avantages : la possibilité de vectoriser un terrain existant, en d'en substituer certaines parties et que ces parties se recollent proprement. D'autres applications de la géométrie différentielles sont aussi envisageables tel que transférer des caractéristiques d'un terrain à un autre.

4.3.3 Formulation Bi-harmonique et expression des contraintes de gradient

L'équation harmonique consiste à appliquer une fois l'opérateur de Laplace,ou Laplacien, à une grandeur, ici l'élévation z, et de résoudre l'équation différentielle de manière à ce que le Laplacien s'annule partout excepté à l'endroit des contraintes. Nous cherchons donc à résoudre pour tout point (x, y), qui n'est pas une contrainte, de la grille de calcul :

$$\Delta z(x, y) = 0 \tag{4.1}$$

Soit :

$$\left(\frac{\partial^2 z}{\partial x^2} + \frac{\partial^2 z}{\partial y^2}\right)(x, y) = 0 \tag{4.2}$$

Par analogie l'équation bi-harmonique est l'équation qui pour tout point (x, y), qui n'est pas une contrainte, de la grille de calcul, vérifie :

$$\Delta\Delta z(x, y) = 0 \tag{4.3}$$

Soit :

$$\left(\frac{\partial^4 z}{\partial x^4} + 2\frac{\partial^2 \partial^2 z}{\partial x^2 \partial y^2} + \frac{\partial^4 z}{\partial y^4}\right)(x, y) = 0 \tag{4.4}$$

Pour utiliser ces équations différentielles dans un solveur il faut les discrétiser. Par soucis de concision nous utilisons les masques matriciels comme notation pour représenter les discrétisations des équations différentielles. Le masque du Laplacien s'écrit :

$$\Delta = \begin{bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{bmatrix} \tag{4.5}$$

Pour trouver le masque correspondant à l'équation harmonique il suffit de convoluer le masque de l'opérateur Laplacien avec lui-même. Ceci nous permet d'obtenir le masque suivant :

$$\begin{aligned} \Delta\Delta &= \begin{bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{bmatrix} * \begin{bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{bmatrix} \\ &= \begin{bmatrix} & & 1 & & \\ & 2 & -8 & 2 & \\ 1 & -8 & 20 & -8 & 1 \\ & 2 & -8 & 2 & \\ & & 1 & & \end{bmatrix} \end{aligned} \tag{4.6}$$

La littérature [74] (section 8.4) nous apprend que résoudre l'équation biharmonique telle-quelle en multi-grille n'est pas efficace, en particulier comparé à l'efficacité de la résolution de l'équation harmonique. Pour obtenir une convergence optimale il faut reformuler la résolution de l'équation biharmonique en la résolution conjointe de deux équations utilisant le Laplacien. Pour cela il suffit d'introduire une variable intermédiaire $w = \Delta z$. L'équation 4.3 devient alors :

$$\begin{cases} \Delta z = w \\ \Delta w = 0 \end{cases} \tag{4.7}$$

Pourtant nous n'avons pas eu besoin d'utiliser cette formulation. Pour pouvoir fixer correctement des contraintes de gradient, nous avons développé le noyau précédant en exprimant des termes de gradient (Équation 4.10). Le but poursuivit est aussi de pouvoir utiliser des contraintes de gradient dans notre système bi-harmonique. Pour faire cela nous cherchons à faire apparaitre les normales comme des propriétés voisines s'exprimant simplement dans l'équation bi-harmonique. L'astuce, pour cela, est de remarquer que l'opérateur bi-harmonique peut-être décomposé en la somme d'un laplacien local et un laplacien moyenné sur un domaine plus grand. La chance est que le laplacien moyenné sur le domaine plus grand peut, lui même, être exprimé en fonction des gradients voisins. On peut alors exprimer les altitudes voisines et les normales voisines dans l'opérateur bi-harmonique.

$$\Delta\Delta = \underbrace{\begin{bmatrix} & & 1 & & \\ & & 0 & & \\ 1 & 0 & -4 & 0 & 1 \\ & & 0 & & \\ & & 1 & & \end{bmatrix}}_{\text{Laplacien des gradients voisins}} - 4 \underbrace{\begin{bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{bmatrix}}_{\text{Laplacien standard}} + \overbrace{\begin{bmatrix} 2 & -4 & 2 \\ -4 & 8 & -4 \\ 2 & -4 & 2 \end{bmatrix}}^{\text{Termes croisé}} \quad (4.8)$$

Le Laplacien de gradient Eq 4.8, peut-être exprimé comme la somme des composantes tangentielles des gradients voisins.

$$
\begin{aligned}
\Delta_{gradient} &= \begin{bmatrix} & & 1 & & \\ & & 0 & & \\ 1 & 0 & -4 & 0 & 1 \\ & & 0 & & \\ & & 1 & & \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & & 1 & & 0 & 0 \\ 0 & & & 0 & & & 0 \\ & & & -1 & & & \\ 1 & 0 & -1 & + & -1 & 0 & 1 \\ & & & -1 & & & \\ 0 & & & 0 & & & 0 \\ 0 & 0 & & 1 & & 0 & 0 \end{bmatrix} \\
&= \begin{bmatrix} & & & 0 & 2d_y grad_y(y+d_y) & & 0 \\ -2d_x grad_x(x-d_x) & & & & & & 2d_x grad_x(x+d_x) \\ & & & 0 & -2d_y grad_y(y-d_y) & & 0 \end{bmatrix}
\end{aligned}
\tag{4.9}
$$

En annulant l'équation bi-harmonique 4.9 nous obtenons l'équation 4.10. Grâce à cette formulation qui explicite les gradients nous pouvons fixer des contraintes de gradient qui sont prises en compte dans l'interpolation bi-harmonique.

$$
\big[z\big] = \frac{1}{4}\begin{bmatrix} & 1 & \\ 1 & 0 & 1 \\ & 1 & \end{bmatrix} - \frac{1}{16}\Delta_{gradient} + \frac{1}{8}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} \tag{4.10}
$$

4.3.4 Contrôle de la forme du terrain

Tandis qu'un solver de Poisson(harmonique/diffusion) génère une surface C^0 qui minimise la variation de gradient, notre solver bi-harmonique génère des surfaces C^1, minimisant la variation de courbure. Or, les terrains que nous souhaitons modéliser contiennent parfois des discontinuités de gradient ; par exemple sur l'arrête d'une montagne, dans le creux d'une vallée, sur les rives d'un lac. Créer de telles discontinuités nécessite alors un traitement spécial.

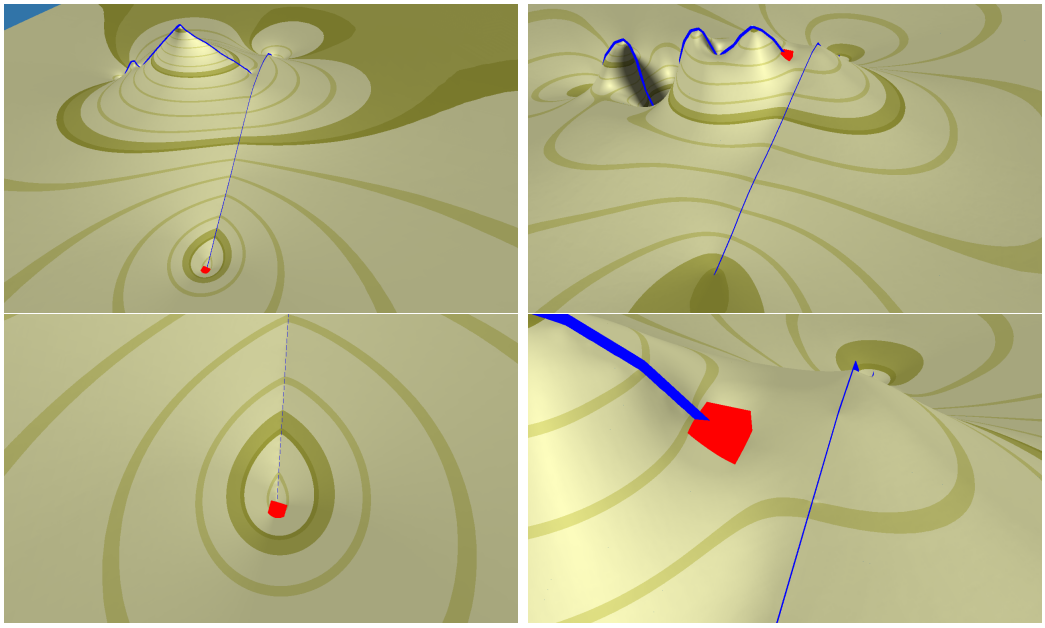

FIGURE 4.2 – Comparaison de deux terrains : l'un généré avec une simple équation harmonique de diffusion (à gauche), l'autre généré avec une équation bi-harmonique (à droite). La continuité C^1 est garantie à l'endroit des contraintes (courbes bleues) dans le cas de l'équation bi-harmonique et non dans le cas de l'équation harmonique. Les lignes de niveaux, plus foncées, sont C^0 à gauche mais C^1 à droite.

Une possibilité serait de relâcher totalement la contrainte de continuité du gradient à l'emplacement de la contrainte. Une autre possibilité serait de fixer de chaque coté le gradient voulu. Nous pensons cependant que ces méthodes ne sont pas pratiques pour la modélisation, c'est pourquoi nous avons adopté une autre approche :

L'utilisateur contrôle le degré de lissage à l'aide d'un paramètre l défini le long des primitives de modélisation. Ce paramètre l est utilisé pour interpoler entre l'équation de diffusion (qui crée des discontinuités à l'endroit des contraintes) et la formulation bi-harmonique. Nous propageons ensuite l à toute la carte par un processus de absorption/diffusion. Le paramètre l peut prendre n'importe quelle valeur positive, de manière à permettre à l'utilisateur de contrôler la largeur de la zone d'influence. Après application du processus d'absorption/diffusion, ces valeurs de l sur le terrain sont divisées par le valeurs de l obtenues par diffusion uniquement, ce qui permet de donner à l une valeur entre 0 et 1.

Une fois calculé, l est alors transformé puis injecté dans l'équation de calcul du terrain, qui devient alors :

$$z_{u,v} = l_t(u,v)z_1(u,v) + (1 - l_t(u,v))z_2(u,v)$$

avec z_1 l'élévation du terrain donnée par l'opérateur bi-harmonique, z_2 celle donnée par l'opérateur harmonique simple et l_t une transformation de l qui permet d'annuler les dérivés de l en $l = 0$ et $l = 1$:

$$l_t = l^2(3 - 2l)$$

Au final notre procédé permet de régler autour de la primitive de contrôle le degré de continuité de la surface, en spécifiant le rayon du lissage.

4.3.5 Édition sur terrain existant

Pour qu'un logiciel soit adopté il est important de pouvoir utiliser ou importer un grand nombre de modèles existants, de manière à ne pas partir d'une feuille blanche et pouvoir se concentrer sur les parties à créer. Pour les terrains, il existe déjà, sous forme de DEM, une base gigantesque de terrains

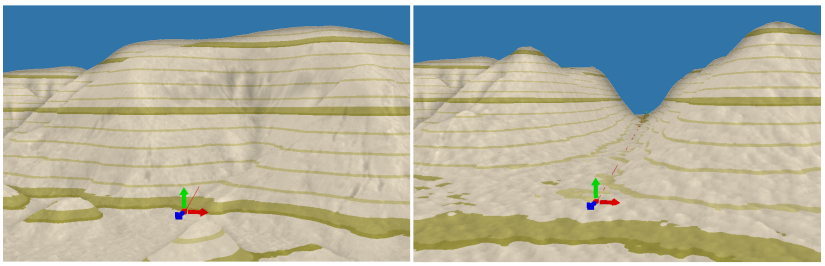

FIGURE 4.3 – Exemple d'édition de terrain réel. A gauche : un terrain réel non modifié. A droite : le même terrain traversé d'une contrainte vectorielle de largeur moyenne. A proximité de la contrainte le DEM est remplacé par un terrain généré qui se recolle au DEM sans discontinuité. Un bruit procédural ressemblant au bruit du DEM est ajouté à la partie générée.

réels. C'est pour cela que nous proposons dans notre logiciel la possibilité d'éditer des terrains existants.

Dans ce cas l'utilisateur peut choisir de charger un modèle l'élévation numérique ou DEM(Digital Elevation Model), qui formera alors la base du terrain édité. Les parties générées du terrain viennent se superposer au DEM grâce à un masque binaire. A l'intérieur des zones il y a génération de terrain vectorielle tandis que le DEM est utilisé à l'extérieur (Figure 4.3).

La formulation bi-harmonique permet de joindre, sans discontinuité, les gradients DEM aux gradients des zones éditables, ce qui permet une jointure lisse entre les zones vectorielles et les zones DEM. Pour cela il suffit d'ajouter des contraintes de gradient au solveur partout ou il y a une frontière entre DEM et vectoriel.

Contraintes de gradient aux frontières : On peut noter un point important concernant l'échantillonnage des contraintes de gradient à la frontière des zones DEM. Avoir des contraintes précises à toutes les échelles de grille permet une bonne convergence du solver multi-grille. Nous avons essayé d'utiliser comme contraintes les gradients fournis par les niveaux de mip-map du DEM. A notre surprise la convergence du système était alors très mauvaise. Au lieu de cela nous échantillonnons les contraintes de gradient au niveaux de la grille la plus fine. Ces contraintes sont alors directement sous échantillonnée par le solveur multi-grille et la convergence est meilleure.

Traitement du bruit procédural : Une seule difficulté persiste : le bruit procédural est ajouté dans les zones vectorielles après que le solveur ait généré le terrain, alors qu'il n'est pas ajouté au DEM. Tel quel, l'ajout du bruit crée une petite discontinuité au niveau de la frontière de recollement. Pour cacher cet artéfact on atténue progressivement le bruit multi-fractal à proximité zones de recollement. Comme les artéfacts visuels dus aux basses fréquences sont plus visibles on choisit d'atténuer les basses fréquences à une distance plus grande que les hautes fréquences.

Génération du masque d'édition : Pour délimiter les zones d'édition nous aurions pu proposer à l'utilisateur de peindre pour sélectionner la zone souhaitée. Cette approche n'aurait pas été élégante vu les efforts que nous déployons pour offrir des outils vectoriels d'édition à l'utilisateur. Au lieu de cela nous permettons à l'utilisateur de définir, un rayon autour des primitives vectorielles. Ce qui se trouve dans le rayon d'influence est interpolé, ce qui est à l'extérieur reste inchangé.

Nous calculons ce champ de distance, de la même manière que le paramètre de lissage l (voir 4.3.4) mais cette fois-ci sans avoir besoin de re-normaliser le résultat. Le résultat est stocké dans l'un des canaux de couleur de la texture résultat.

Limitations : Nous pensons que ce type d'interaction est pratique pour les personnes souhaitant définir une zone d'édition relative à leurs primitives d'édition, de sorte que quand ils déforment celles-ci, la zone d'interpolation accompagne cette déformation. L'utilisateur souhaite parfois définir précisément les frontières d'interpolation, ce que cette méthode ne permet pas directement. Il est tout de même facile de contourner cette limitation. Il suffit pour cela de poser des contraintes de zone de valeur nulle à l'endroit des frontières voulues et de compléter ces contraintes par une contrainte de valeur positive à l'intérieur de la zone. Bien que fonctionnelle, cette méthode ne fait que contourner une limitation.

4.4 Modélisation Interactive

La création de terrains en temps réel implique de savoir traiter des données qui changent constamment et dont la taille augmente exponentiellement avec la résolution du terrain. Dans cette partie nous décrivons comment obtenir un système de modélisation interactif, capable de supporter des grandes quantités de données.

Comme on l'a dit précédemment le solveur multi-grille est efficace et est capable de générer un nouveau terrain de 16Mpx à une fréquence de 20 images par seconde avec une carte graphique nvidia gforce 460. Pour donner une idée la quantité de données générées correspond à une bande passante de 10Gb/s, comparé aux 8Gb/s que supporte un port PCI-express 2.0 16x. Comme notre but est la modélisation interactive il faut que le cycle rendu-interaction-calcul soit très rapide. De tels volumes de données nous obligent à optimiser nos algorithmes, en particulier les transferts CPU/GPU. De plus si l'on souhaite obtenir un affichage de qualité du terrain quelque soit le point de vue il faut que la géométrie soit adaptative et puisse se mettre à jour rapidement.

Dans notre système, deux types d'actions nécessitant des calculs peuvent impacter les performances. La première est une action nécessitant une mise à jour du terrain ; ce qui a pour conséquence de déclencher l'exécution de la chaine entière de génération de terrain. La seconde action est la mise à jour du rendu du terrain. Cette action est exécutée juste après la génération de terrain ou quand la caméra change de position. Chacune de ces actions fait l'objet d'une sous partie. La figure 4.4 explique comment CPU et GPU se répartissent le travail. Cette figure sera mieux détaillée dans les sous-parties suivantes.

4.4.1 Mise à jour du terrain

La mise à jour du terrain est couteuse en temps de calcul, c'est pourquoi nous avons essayé de l'optimiser. La figure 4.5 décrit le fonctionnement global du processus de génération de terrain.

La première étape consiste à transformer les primitives d'édition en contraintes pour le solveur multi-grille (voir partie gauche de la Figure 4.5). Cela

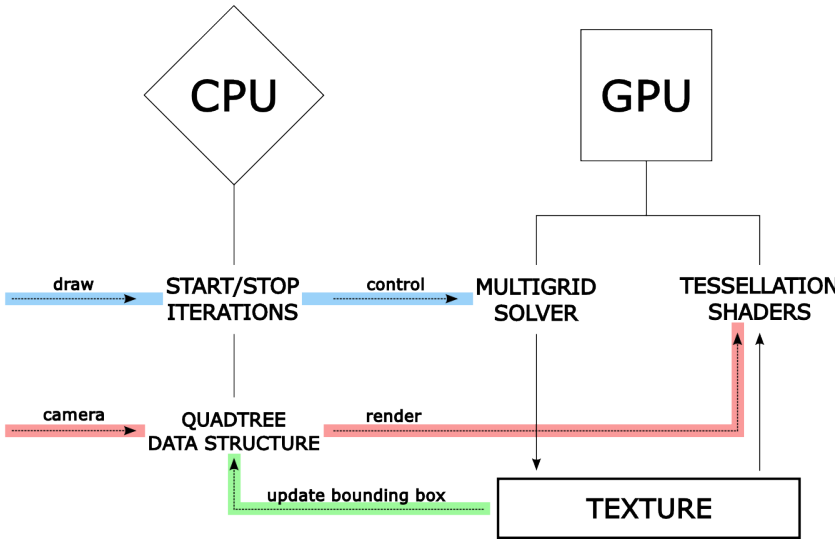

FIGURE 4.4 – Chaine de traitement de notre système. Alors que le CPU contrôle le solveur multi-grille en fonction des actions utilisateur, le GPU génère le terrain dans une texture et effectue la tesselation de la carte d'élévation qu'il vient de produire. Il faut pour cela : contrôler les itérations du solveur (bleu), faire un rendu adaptatif du résultat (rouge), Mettre à jour la géométrie grossière contenue sur CPU (vert).

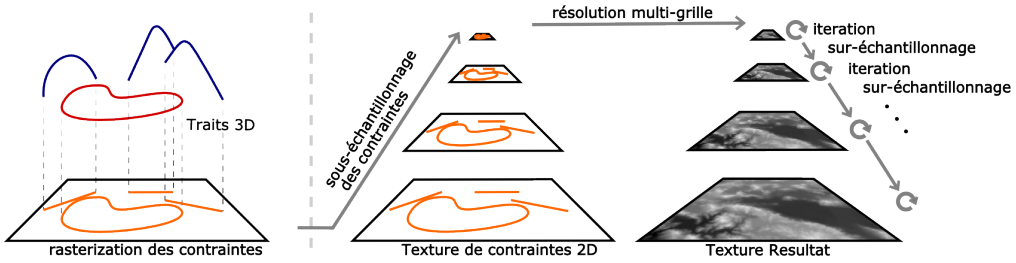

FIGURE 4.5 – Schéma d'ensemble de la génération de terrain.

revient à dessiner ces contraintes dans une texture dite texture de contraintes. Ces dernières sont contenues dans un graphe de scène et sont passées à OpenGL, au moment de leur création, sous forme de VBO. Lorsque qu'une primitive est éditée seul son VBO est modifié. Dessiner la texture de contraintes demande alors aucun transfert GPU supplémentaire. La texture de contraintes est alors sous échantillonnées en grilles de tailles chaque fois deux fois plus petites (voir milieu de la Figure 4.5).

Le processus de résolution multi-grille commence alors et s'effectue en deux passes. Ces deux passes fonctionnent de la même manière (voir partie droite de la Figure 4.5). La résolution commence au niveau le plus grossier des textures puis s'applique à tous les niveaux de texture, l'un après l'autre. A chaque fois un nombre donné d'itérations de Jacobi est appliqué à la texture. Une fois les itérations appliquées la texture résultat est sur-échantillonnée pour initialiser la résolution au prochain niveau de résolution. Le processus se répète alors jusqu'à ce que l'on atteigne le dernier niveau de texture. Le nombre d'itérations dépend de la précision souhaitée du système mais influence directement la vitesse du solveur.

La première passe permet faire la résolution des paramètres de lissage, du masque d'édition ainsi que des paramètres de bruit. La texture générée sert alors de seconde texture de contraintes pour effectuer cette fois-ci la résolution de la carte d'élévation.

Le résultat de cette étape est un terrain généré, stocké sous la forme d'une texture mip-mapée.

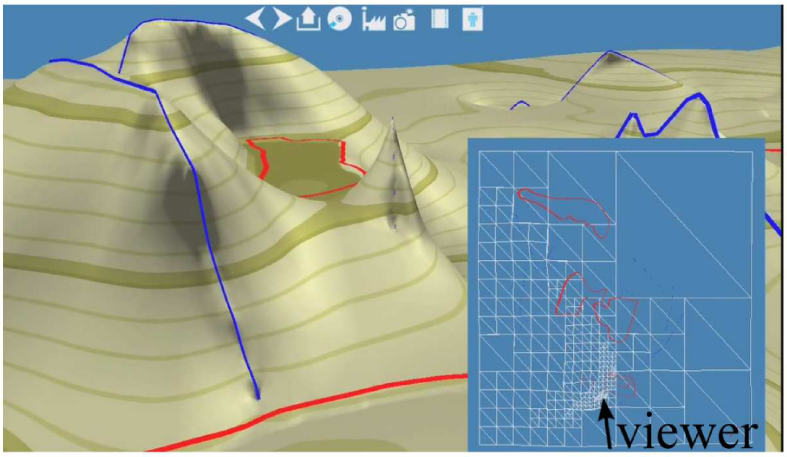

FIGURE 4.6 – L'encart à droite montre une vue aérienne du maillage grossier utilisé pour l'affichage. On observe que le quad-tree est plus raffiné à proximité de la caméra, mais que les carreau qui n'apparaissent pas à l'écran, sont peu subdivisés

4.4.2 Affichage du terrain

L'affichage du terrain se décompose en deux parties sur CPU et sur GPU.

Contrôle du Quad-Tree sur CPU : Dans un premier temps, un maillage grossier du terrain est maintenu coté CPU à l'aide d'un quad-tree, dont les feuilles sont des carreaux de terrain (quad) dont les plus proches de la caméra sont les plus subdivisées (voir Figure 4.6). Cette petite structure de données permet de contrôler le rendu sur CPU en n'envoyant que les quads nécessaires au GPU (partie rouge de la Figure 4.4).

Tesselation sur GPU : Dans un second temps une version précise du terrain est générée à l'aide des deux parties programmables de tesselation présentes sur les GPU modernes : la partie de contrôle et celle d'évaluation. La partie de contrôle est chargée de subdiviser régulièrement chaque carreau de terrain, c'est à dire les feuilles du quad-tree maintenu sur CPU, dont la résolution peut-être augmentée dans la limite des capacités de tesselation du GPU. Notre algorithme utilise une subdivision uniforme définie globalement

au niveau de l'application. La partie d'évaluation, quant à elle, définie les propriétés de chaque sommet généré lors de la tesselation. Dans notre cas, le programme va lire dans la texture de terrain, générée par le solveur, les élévations nécessaires au positionnement de chaque sommet (partie bleue de la Figure 4.4).

Des jonctions en T peuvent apparaitre lorsque deux patchs adjacents ont un niveau de subdivision différent. En pratique nous n'essayons même pas de traiter ce problème car les triangles générés par la tesselation sont si fins que l'artéfact est imperceptible. Néanmoins la partie de contrôle de la tesselation permettrait d'imposer que les bords des patchs, voisins d'un patch moins subdivisé, soient deux fois moins subdivisés. Cette solution suffirait à résoudre ce problème. Pour finir avec la tesselation, nous utilisons une subdivision régulière des patchs mais il est possible d'implémenter un mécanisme plus fin de niveaux de détail. La subdivision régulière nous était suffisante.

Une fois la tesselation effectuée nous utilisons un algorithme de rendu ad-hoc très simple qui prend en compte une composante de lumière directionnelle type soleil, une composante de lumière ambiante et utilise l'élévation z pour dessiner procéduralement des lignes de niveaux.

Contrôle du niveau de détails (LOD) : Lorsque le terrain est modifié ou que la caméra bouge il faut mettre à jour le quad-tree ; pour cela nous utilisons un mécanisme de niveau de détails. Notre méthode projette sur l'écran les boites englobantes des patchs de terrain pour savoir si ces derniers doivent être subdivisés ou non (voir Figure 4.7). Nous utilisons une mesure en pixels, de la largeur et de la hauteur de ces boites englobantes, pour subdiviser récursivement le quad-tree, de manière à ce que le quad-tree s'adapte à un redimensionnement du viewport. Pour cela nous calculons la boite englobante 2D de la projection, sur le viewport, de la boite englobante 3D.

Calcul des boites englobantes : Calculer les boites englobantes sur CPU nécessiterai beaucoup de calculs et surtout de rapatrier le terrain du GPU vers le CPU. Au lieu de cela nous calculons les boites englobantes sur GPU de manière hiérarchique, en stockant les hauteurs minimales et maximales contenues dans un patch. Nous ne renvoyons sur CPU que les boites englobantes ayant une taille comparable aux feuilles du quad-tree qui sont elles

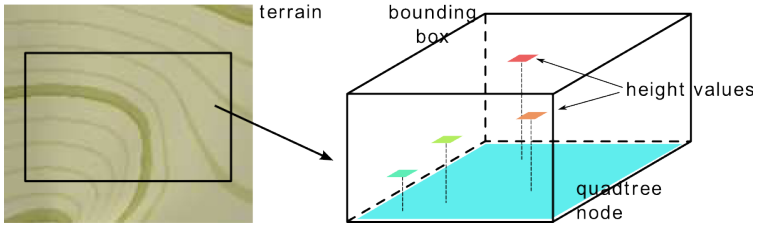

FIGURE 4.7 – Illustration d'un carreau de terrain et de sa boite englobante. Les altitudes minimales et maximales sont utilisées pour calculer l'extension verticale de la boite englobante. Ce boites sont ensuite projetée sur le viewport pour guider la subdivision du quad-tree.

très grossières (partie verte de la Figure 4.4).

Cette solution répartit la génération de la géométrie entre CPU et GPU : La géométrie adaptative est générée grossièrement sur CPU puis massivement raffinée sur GPU à l'aide de la tesselation.

4.5 Interface éditeur de terrains

L'interface a été une préoccupation constante de notre travail. C'est pour cette raison que nous avons choisi de la génération en temps réel de terrain, que nous avons cherché un contrôle meilleur des gradients et pour cette raison encore que nous avons souhaité que l'utilisateur puisse éditer le monde, vu de la première personne, en 3D, pour rendre l'édition plus immersive. Nous avons pour cela créé des primitives vectorielles 3D qu'il est possible de modifier à la première personne.

Nous présentons ci-dessous l'interface 3D que nous avons implémentée, avant d'expliquer comment placer des primitives pour vectoriser un terrain existant.

4.5.1 Manipulation 3D directe

Le premier point d'interface important est la manipulation 3D directe du terrain. Pour cela nous avons créé des primitives géométriques 3D : des points, lignes, courbes de Bézier. Ces primitives sont placées dans la scène 3D, bien

qu'elles doivent être projetées en 2D pour être transmises au solveur multigrille.

Les points de contrôle sont des éléments qui peuvent avoir des données supplémentaires attachées : les paramètres de bruit, de masque, de lissage, de normale, de hauteur. Tout point de contrôle peut avoir ces attributs définis ou pas. Si bien que certains points de contrôle peuvent ne servir qu'au contrôle du bruit procédural, tandis que d'autres peuvent servir à contraindre simultanément altitude, bruit, normales.

Pour réussir une implémentation correcte d'interface 3D il faut séparer les widget, leur manipulation, le rendu des widgets, et enfin le rendu des contraintes dans le solveur. Cette approche permet de contrôler indépendamment : rendu, interface utilisateur et génération de terrain.

Dans notre implémentation, le rendu 3d et la génération de terrain sont gérés grâce à des primitives géométriques OpenGL définies au sein de la bibliothèque Ogre3D. Ces primitives, qui sont utilisées pour l'affichage et pour dessiner les contraintes pour le solveur ,appartiennent à des graphes de scène séparés. Quand on souhaite mettre à jour le terrain il faut alors redessiner les primitives de contraintes puis relancer le processus de génération de terrain. Si une modification n'affecte que le rendu alors il suffit juste de redessiner la scène.

Une extension logique de ce travail serait l'élaboration de primitives spécifiques pour la création de falaises, de rivières, de crêtes de montagne. De cette manière, l'utilisateur pourrait utiliser des outils adaptés aux types de reliefs qu'il désire créer. Cependant nous n'avons pas eu le temps d'implémenter ces fonctionnalités.

4.5.2 Vectorisation d'un terrain existant

Un point remarquable de notre méthode est qu'il est non seulement facile d'intégrer l'édition au sein d'un terrain existant (Section 4.3.5), il est aussi facile de le vectoriser à l'aide de primitives bien placées. La difficulté pour vectoriser efficacement un terrain est justement de savoir comment bien placer les contraintes pour en minimiser le nombre. C'est à dire comment minimiser l'erreur entre le terrain réel et le terrain vectorisé tout en minimisant le nombre de contraintes crées pour vectoriser le terrain.

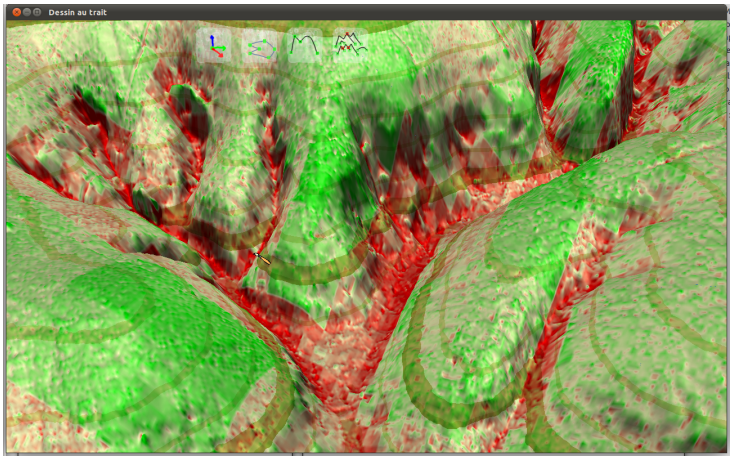

FIGURE 4.8 – Affichage d'un guide de vectorisation. Le rouge indique les zones concaves et le vert indique les zones convexes. En plaçant les contraintes aux endroits de couleur vive on obtient une reconstruction de bonne qualité, ce qui semble assez intuitif.

Considérons un terrain sous la forme d'une carte d'élévation (DEM). Quels sont les points de ce terrain qui ne sont pas intéressants en tant que contraintes ? Ce sont les points qui peuvent être calculés par interpolation, c'est à dire les points dont le laplacien/bi-laplacien est nul.

En partant de ce principe, nous avons essayé de créer des guides de vectorisation qui indiquent ou placer les contraintes vectorielles. Nous évaluons le bi-laplacien à différents niveaux de résolution de manière à atténuer le bruit résiduel du DEM, puis nous sommons toutes ces contributions. Enfin nous affichons sur le terrain la somme de ces contributions (Figure 4.8).

On observe que les zones où il faut placer des contraintes semblent plutôt intuitives : ce sont les zones où la courbure varie significativement. Pour cette raison il est facile, même sans guide, de placer les contraintes vectorielles. D'autant qu'aucun paramétrage spécifique n'est nécessaire : il suffit de donner aux contraintes, les valeurs d'altitude et de gradient des contraintes observés sur le terrain réel.

4.6 Conclusion et Perspectives

Nous avons présenté dans ce chapitre un modeleur de terrain temps réel, contrôlable et capable de supporter des terrains de grandes tailles. Ce travail est intéressant car il forme de notre point de vue une base saine pour le développement d'autres modeleurs de terrain.

Nous avons toutefois rencontré pas mal de difficultés, sur de petits détails, pour que le solveur multi-grille fonctionne comme nous le souhaitions.

Parmis les perspectives générées par ce travail beaucoup se basent sur l'équation bi-harmonique. Nous aimerions par exemple l'utiliser pour pour essayer de construire un modeleur qui autorise la modélisation volumique par squelette ainsi que la modélisation surfacique. De plus on pourrait essayer d'appliquer le même principe à des fonctions de champ, pour faire une sorte de modélisation implicite différentielle. Ce serait une piste de recherche très originale, que nous n'avons pas encore observé.

4.6.1 Extension à des surfaces libres

L'avantage de notre algorithme par rapport à un algorithme plus spécifique aux terrain est qu'il peut être facilement étendu à l'édition interactive adaptative de surfaces autres que des terrains représentés par des cartes d'élévations. En effet on peut stocker des positions au lieu d'élévations dans les textures situées sur le GPU et ne renvoyer que les boites englobantes des positions contenues dans une tuile. Cette simple modification permet de passer du rendu du terrain au rendu de surfaces arbitraires. Le genre topologique de la surface ainsi créée peut-être changé au moment de la création du modèle.

Chapitre 5

Modélisation de terrains par croquis

5.1 Résumé

Dans ce chapitre nous présentons un système de modélisation de montagnes à base de croquis. L'utilisateur peut dessiner un paysage constitué de multiples traits de silhouettes qui est analysé grâce à des connaissances apriori. Cette analyse nous aide à positionner les silhouettes dans l'espace. Le terrain est enfin généré en utilisant un algorithme d'interpolation, l'algorithme présenté dans le chapitre précédent. Notre système a été conçu pour pouvoir éditer le résultat de la génération en 3D en temps réel, sans perdre la possibilité de compléter, de modifier le dessin. L'analyse, la génération et l'édition temps réel en 3D du terrain que nous présentons ici permettent de simplifier grandement la tache ardue qu'est la modélisation d'un paysage depuis un point de vue fixe en vue perspective.

5.2 Introduction

De nos jours le rendu et la manipulation de modèles complexes n'est plus un problème, car nos ordinateurs sont suffisamment performants pour gérer cette complexité. La génération de terrain pour la création de paysages virtuels

reste toutefois difficile : les méthodes de génération existantes (vue dans l'état de l'art en 1.1.2) vont de la génération fractale de paysage, qui offre un très beau résultat mais très peu de contrôle, aux méthodes de design interactif consistant à "peindre" une carte d'élévation en vue de dessus. Un aspect peu abordé mais pourtant essentiel pour la modélisation de paysages est la possibilité de "dessiner" un terrain en vue perspective. En effet, quand une personne souhaite représenter un paysage, elle prend un papier et un crayon, et dessine les silhouettes de collines ou de montagnes qu'elle observe depuis son point de vue, en représentant dans un ordre indéterminé des relief proches (monticules) comme lointains (chaines montagneuses). Effectuer ce dessin de notre propre point de vue est naturel, alors que contrôler l'aspect et le positionnement du terrain à l'écran, vu de la troisième personne, est difficile avec les méthodes citées existantes.

Dans ce chapitre nous explorons la création de terrains à partir d'une esquisse représentant ses silhouettes vues d'une position de caméra donnée. On présente à l'utilisateur une interface par croquis dans laquelle il a la liberté de dessiner ces silhouettes dans un ordre arbitraire. Les traits de silhouettes (un graphe de lignes présentant des embranchements, dus aux occultation des parties du terrain par d'autres) sont ensuite interprétés pour calculer l'ordre des silhouettes, ainsi que leurs relations. Ces informations sont utilisées pour générer un positionnement en 3D plausible pour ces courbes ensuite utilisées comme contraintes. Ces traits de silhouettes, placés en 3D, sont enfin utilisés dans l'algorithme d'interpolation du chapitre 4, qui génère le maillage du terrain. En plus de cela l'utilisateur peut ajuster la position des silhouettes depuis un autre point de vue tout en conservant la projection, c'est à dire la concordance avec le dessin initial. Il peut enfin continuer/modifier son dessin depuis le point de vue initial, sans perdre les ajustements effectués. Comme toutes ces étapes sont très rapides, l'analyse, la projection et la génération de terrain peuvent se faire en temps réel.

Ce chapitre se décompose en trois grandes parties. La première traite de l'analyse du croquis de l'utilisateur. La seconde explique comment nous générons le terrain à l'aide des information extraites du croquis. Enfin la dernière partie explique l'interface que nous avons choisi d'implémenter.

5.3 Interprétation de silhouettes de terrain

Notre système est un outil de modélisation par croquis. L'utilisateur peut utiliser une tablette graphique ou une souris pour saisir des courbes 2D qui correspondent aux silhouettes du terrain qu'il souhaite modéliser. Nous qualifions notre système de multi-trait car au lieu de reconstruire les traits de silhouette les uns après les autres comme cela est fait dans [37][77][44] nous avons pris le parti d'interpréter un croquis constitué de multiples trait de silhouettes comme dans [46][47]. L'avantage de considérer un croquis composé est que l'on peut analyser les relations entre les traits pour en extraire plus informations qu'avec un seul trait. L'inconvénient est qu'il est souvent complexe voir impossible d'interpréter un croquis à cause de la diversité d'interprétations qui peuvent exister.

Dans cette section nous utilisons des connaissances apriori sur les terrains pour simplifier drastiquement l'interprétation des relations entre les traits de silhouettes. Ces simplifications nous permettent de trier les silhouettes par rapport à leur profondeur en complexité sous-quadratique/linéarithmique. De même nous caractérisons les relations qui relient les extrémités de chaque silhouette au reste du dessin.

5.3.1 Ordonnancement des traits de silhouettes

Pour structurer le croquis on cherche à déterminer quelle silhouette se trouve devant quelle autre, c'est à dire ordonnancer les traits de silhouettes. Pour cela il convient de commencer par observer les silhouettes de paysages réels. La première observation que l'on peut faire est que les silhouettes de terrain comportent rarement des surplombs. Et en l'absence de surplombs, une silhouette de montagne ne se recouvre pas quand on la projette sur la ligne d'horizon. Comme le terrain que nous générons est une carte d'élévation on peut, sans trop perdre en généralité, ignorer les surplombs. Cette propriété de projection injective de la silhouette sur l'horizon permet de réduire l'ordonnancement des silhouettes à un balayage gauche/droite de l'écran. Si l'on observe maintenant les silhouettes, perpendiculairement à la ligne d'horizon on peut affirmer que si une silhouette de situe au dessus d'une autre c'est qu'elle est plus éloignée.

L'algorithme que nous proposons pour trier les silhouettes consiste donc à projeter les traits sur la ligne d'horizon, de manière à pouvoir les balayer de gauche à droite. De cette manière on réduit la dimensionnalité de notre problème à un. Durant le balayage, à l'abscisse x, on considère à chaque moment la liste des traits présents en x (Fig 5.1, partie basse). Quand un nouveau trait est inséré dans la liste on regarde quels traits se trouvent au dessus et en dessous de lui et on l'insère entre les deux. Pour balayer, l'opération la plus couteuse est de trier par abscisse croissante les extrémités gauches et droite des silhouettes au début de l'algorithme. Ensuite chaque insertion de nouvelle silhouette dans la liste triée se fait en temps logarithmique. L'algorithme est donc particulièrement efficace.

Avant de lancer l'algorithme de balayage on filtre les silhouettes pour supprimer les surplombs, ré-échantillonner et segmenter (Fig 5.1, partie haute). La segmentation sert à introduire des points de contrôle qui pourront être manipulés par l'utilisateur.

5.3.2 Étiquetage des relations

Ordonner les silhouettes en fonction de leur profondeur n'est pas suffisant pour positionner les silhouettes en 3D. Pour cela nous devons aussi quantifier la différence de profondeur entre les silhouettes. Les extrémités des silhouettes nous permettent de faire cela selon qu'elles sont occultées ou reposent sur d'autre silhouettes.

Occultation : L'occultation d'une silhouette par une autre s'observe facilement au niveau de l'extrémité de la silhouette occultée car elle fait apparaitre une arête en T (Figure 5.1, relation de la troisième ligne), la patte du T pointant vers le haut. Grâce à l'occultation nous pouvons connaitre l'ordre entre deux primitives ; l'occultée se trouve derrière. Quantifier la distance qui sépare les deux silhouettes est difficile car elles peuvent être très proches comme très éloignées.

Naissance d'une silhouette : A l'opposé si une extrémité de silhouette n'est pas occultée cela veut dire que cette silhouette nait en ce point du terrain. Nous pouvons déduire que la profondeur de l'extrémité est égale à

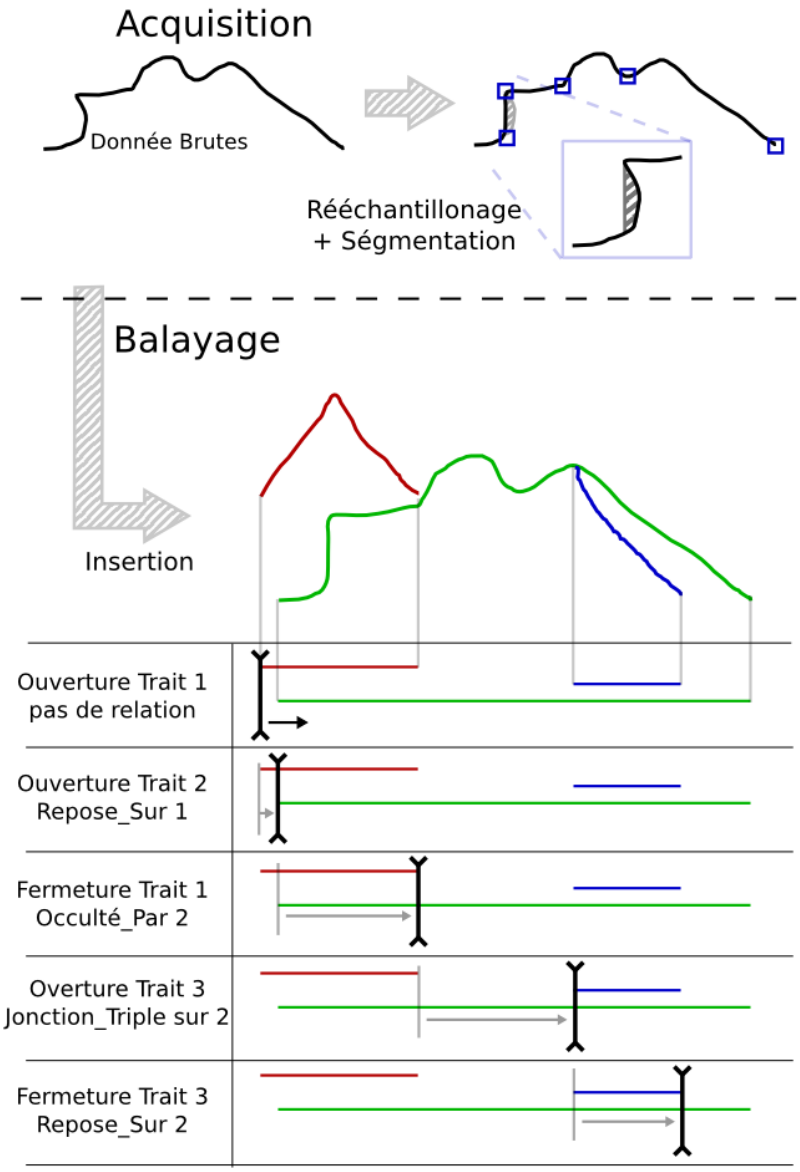

FIGURE 5.1 – Algorithme de balayage

la profondeur du terrain en ce point. La silhouette qui se trouve directement au dessus de l'extrémité est alors la silhouette qui se trouve juste derrière. Pour ce qui est de quantifier la distance séparant les silhouette arrière de l'extrémité nous pouvons faire des hypothèses sur la forme de la silhouette arrière. Cela est expliqué dans la partie *Génération de la géométrie*.

Trois types de relations et leur calcul fiable pendant la balayage : Les extrémités peuvent être étiquetées de 3 manières différentes : OCCULTÉE_PAR, REPOSE_SUR et JONCTION_TRIPLE. Le calcul des relations se fait pendant le balayage, au moment de l'ajout ou du retrait d'une extrémité. À cet instant nous connaissons la liste des traits présents au niveau de cette abscisse. Si l'ordonnée est trop éloignée de tout autre silhouette alors on considère que c'est une silhouette naissante : REPOSE_SUR. Si l'ordonnée de notre extrémité est proche d'une silhouette présente on regarde la silhouette un peu plus loin pour voir si elle part au dessus ou en dessous. Si elle part au dessus alors c'est une relation OCCULTÉE_PAR, sinon c'est une relation JONCTION_TRIPLE. Regarder un segment plutôt que ponctuellement rend la détection plus stable.

La partie droite de la figure 5.1 montre un exemple de balayage et illustre la détection des types de relation.

5.4 Génération de géométrie 3D

Dans la section précédente nous avons présenté comment nous analysons un croquis de paysage. Nous avons extrait d'une part l'ordre en profondeur des traits de silhouette et d'autre part des informations concernant leurs extrémités. Dans cette section, nous présentons la génération du terrain à partir du croquis interprété. Pour cela nous commençons par donner une troisième dimension aux traits de silhouettes. Dans un deuxième temps nous utilisons ces silhouettes 3D comme contraintes dans le générateur de terrain du chapitre 4, créant ainsi un terrain interpolant les silhouettes 3D. L'ajout de la troisième dimension, que nous appelons dé-projection, et la génération de terrain sont expliquées ci dessous.

5.4.1 Inférence de la 3D à partir de silhouettes 2D et de connaissances a priori

Pour comprendre comment l'inférence de la 3D se fait, il faut savoir que l'utilisateur ne part pas de rien. Il dessine en effet sur un terrain déjà existant, simple comme un plan ou bien plus complexe. Ce terrain existant forme une base sur laquelle les traits de silhouette 2D viennent se projeter. Cette projection constitue la distance maximale à laquelle les silhouettes peuvent être projetées. Pour cela, la première étape de notre algorithme de positionnement consiste à projeter au plus loin les silhouettes en commençant par les silhouettes qui se trouvent à l'arrière. A chaque nouvelle silhouette projetée on vérifie que la projection est conforme aux connaissances extraites du croquis. C'est ce que l'on appelle la projection au plus loin. La seconde étape, nommée ajustement au plus près, consiste à rapprocher les silhouettes qui ont été projetées trop loin. C'est généralement le cas d'une silhouette qui occultée en ses deux extrémités.

Projection au plus loin : Lorsque l'on projette au plus loin une silhouette on s'assure d'abord que les silhouettes se trouvant derrière sont projetées en premier. Une silhouette peut se retrouver projetée soit sur la silhouette qui se trouve derrière elle, soit sur le terrain. Ces deux options sont parfois possible, on choisi alors celle pour laquelle la silhouette projetée est la plus proche de la caméra (Figure 5.2). Projeter sur le terrain existant est facile puisque qu'il existe. Il est moins facile de projeter sur une silhouette car elle n'est associée à aucun terrain. Pourtant en regardant une silhouette un humain peut imaginer le relief lui correspondant. Cela veut dire qu'une connaissance apriori peut être utilisée pour estimer le relief à partir de sa silhouette (Fig 5.2). Nous considérons par exemple ici que le profile de la montagne doit être semblable à une parabole. Évidement cela suppose que la silhouette arrière est déjà positionnée, c'est pour cela que nous appliquons la *projection au plus loin* d'arrière vers l'avant.

Enfin on doit noter que des silhouettes ne reposant sur rien peuvent être projetées à l'infini, elles seront rapprochées dans l'étape suivante.

Ajustement au plus près : Les extrémités qui sont occultées ne reposent pas sur un terrain, nous savons seulement que leur profondeur est supérieure à

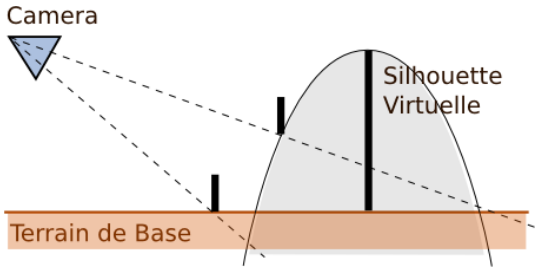

FIGURE 5.2 – Projection au plus loin

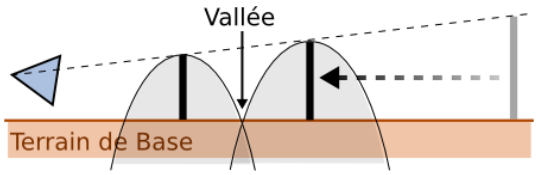

FIGURE 5.3 – Ajustement au plus près

la profondeur de la silhouette qui les occulte. Comme dans l'étape précédente les silhouettes ont été "projetées au plus loin", nous souhaitons maintenant rapprocher les silhouettes occultées trop distantes. Pour autant nous ne voulons pas trop les rapprocher ; pour cela nous supposons qu'une vallée sépare nos deux silhouettes. Nous rapprochons donc la silhouette en garantissant qu'une vallée existe devant celle-ci. La figure 5.3 explique de nouveau comment une supposition sur la forme des montagnes nous permet d'estimer la distance minimale qui garanti la présence d'une vallée.

Une fois que les primitives occultées ont étés rapprochées on applique de nouveau *projection* et *ajustement* pour mettre à jour les positions des dernières primitives. On fait ici toujours attention à ne pas trop avancer les silhouettes pour conserver l'ordre des primitives.

A la fin de cette étape, toutes les silhouettes dessinées par l'utilisateur ont été converties en courbes 3D.

5.4.2 Génération de terrain à partir de Silhouettes 3D

Pour générer le terrain nous utilisons le système présenté dans le chapitre 4. Nous transformons les silhouettes 3D, calculées précédemment, en con-

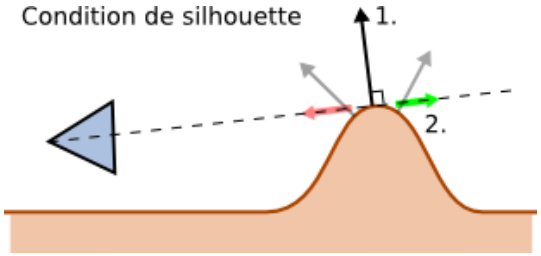

FIGURE 5.4 – Contraintes pour obtenir des silhouettes

traintes d'interpolation, qui servent à guider la génération de terrain. Nous souhaitons que le terrain généré passe par les contraintes 3D, mais aussi que le terrain généré présente des silhouettes au niveau de ces contraintes. Géométriquement pour qu'un point du terrain soit une silhouette il faut que la normale du terrain en ce point soit orthogonale à la direction de la caméra. Il faut aussi que le produit scalaire de la normale avec la direction de la caméra change de signe au niveau de la silhouette (Fig 5.4). Bien que le système multi-grille présenté soit capable de garantir la première contrainte mais pas la seconde, dans la majorité des cas cette contrainte suffit à obtenir des silhouettes là ou nous le souhaitons.

Nous avons donc défini une nouvelle primitive d'édition de terrain qui est la "SilhouetteDeMontagne". Cette primitive projective est mieux expliquée dans la section suivante. Ce qu'il faut noter ici est que la contrainte de gradient le long de la silhouette est orthogonale à la caméra ainsi qu'a la silhouette elle même. Cette contrainte est définie comme :

$$n_P = t(\mathbf{P}) \times \frac{\mathbf{CP}}{\|\mathbf{CP}\|}$$

Avec $\mathbf{C}$ la position de la caméra, $\mathbf{P}$ un point de la silhouette, $t(\mathbf{P})$ la composante tangentielle de la silhouette au point $\mathbf{P}$ et n_P la normale souhaitée du terrain au point $\mathbf{P}$.

5.5 Édition Interactive du Terrain

En plus de proposer une reconstruction automatique à partir d'un croquis multi-traits, le système que nous présentons permet d'ajuster la profondeur de chacun des traits en maintenant leur projection sur le dessin , tout en permettant aussi la correction du dessin initial. C'est cela que nous présentons ici.

5.5.1 Ajustement depuis un point de vue différent qui maintient la projection

En sus de l'interprétation des croquis et de la génération de terrain nous souhaitons développer la créativité et l'efficacité de la modélisation à partir du croquis. Le fait de pouvoir modéliser un terrain à partir d'un point de vue unique est important. Seulement la dé-projection choisit une solution plausible parmi une infinité de possibilités, et le résultat peut ne pas être celui souhaité. Pour palier ce problème l'utilisateur doit pouvoir ajuster le résultat de la dé-projection depuis un autre point de vue ; c'est à dire les profondeurs des points de contrôles de chaque silhouette. Nous offrons cette fonctionnalité grâce à des déplacements contraints des courbes 3D. Ce faisant le terrain est mis à jour continument durant l'ajustement des points, ce qui permet à l'utilisateur d'observer le résultat à chaque instant.

5.5.2 Édition du croquis : mécanisme d'interprétation tardive

Contrairement aux approches précédentes de modélisation par croquis nous souhaitons permettre une modification conjointe 2D via croquis et 3D.

En effet un aspect de l'ergonomie est la manière dont les opérations d'éditions peuvent s'enchainer. Le flux de travail, comme nous l'appellerons ici, est particulièrement fluide quand il s'agit de croquis sur papier, car les actions de dessin et de gommage peuvent être effectuées dans n'importe quel ordre. Cela permet à l'artiste d'exprimer sa créativité sans contraintes. En ce qui concerne la modélisation numérique l'ordre des opérations est souvent

essentiel et rigide, ce qui contribue à réduire la créativité. Si les interfaces par croquis sont souvent utilisée en création numérique pour fournir un contrôle plus intuitif, certains systèmes contraignent l'ordre des opérations. Or, l'ajout d'un détail peut changer radicalement l'interprétation d'un dessin. Si le dessin a déjà été analysé, la géométrie générée et des modifications effectuées il est difficile de pouvoir réinterpréter le dessin original. Réinterpréter, c'est pourtant ce que notre cerveau fait à chaque ajout de trait dans un croquis.

Nous souhaitons rendre possible la réinterprétation dans notre application. Pour cela nous avons implémenté ce que nous appelons une interprétation tardive du croquis. La figure 5.5 illustre ce principe qui peut se résumer en une phrase : Les éditions 3D sont des éditions 2D directement dans le croquis. Tant que l'on continue à travailler directement sur le croquis on peut conserver les modifications 3D, même partiellement, et autoriser une modification du croquis.

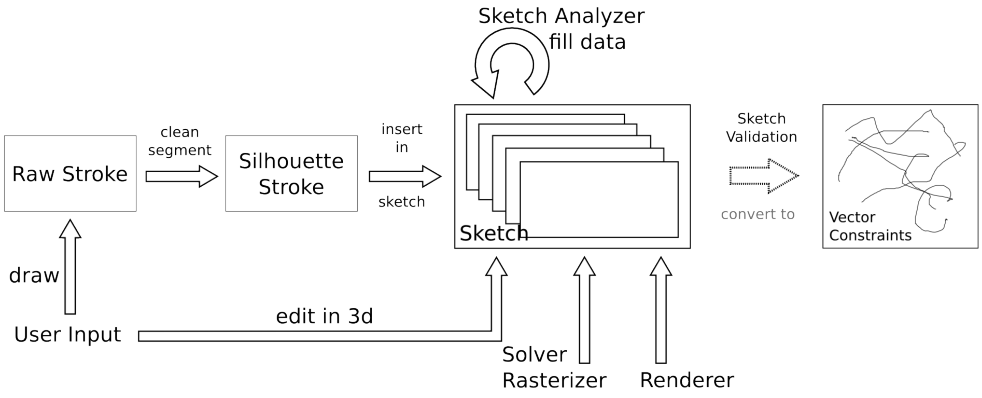

FIGURE 5.5 – Interprétation Tardive du dessin

Grâce à cette approche, l'utilisateur est libre d'alterner entre édition 2D du croquis et 3D directement dans le monde. Nous pensons que cette cohérence favorise la créativité car donne plus de liberté.

5.6 Résultats

L'analyse du croquis, la création des contraintes et la génération de terrain prennent en temps environs 50ms. Ce qui permet la continuelle réinterprétation du croquis si nécessaire. La résolution du terrain généré est de (4k x 4k). Ces performances sont les mêmes que celles observées dans le chapitre précédent.

Notre système constitue une tentative intéressante de créer un système de modélisation multi-traits, de montagnes, à la première personne.

Du point de vue de l'interface le fait que le système laisse la liberté à l'utilisateur d'alterner entre 2D et 3D ainsi que de dessiner plusieurs traits d'un coup donne des qualités certaines au système. Le positionnement automatique de montagnes fonctionne relativement bien, bien qu'il souffre de petites instabilités. En particulier l'algorithme de positionnement peut placer devant une petite montagne qui devrait être derrière lorsqu'une montagne est déjà trop proche derrière.

La partie du travail qui pourrait être améliorée est la reconstruction en elle-même. Il n'y a par exemple pas de contraintes explicites pour créer les vallées. Il se peut alors que deux montagnes trop proches l'une de l'autre ne soient pas séparées par une vallée. Ce que l'on voit de la montagne de devant n'est alors pas une silhouette. L'autre lacune de la génération de terrain est l'analyse individuelle des silhouettes. Les silhouettes sont lisses par défaut, à moins que l'un des points de segmentation soit une pointe vers le haut. Cette interprétation est souvent trop simpliste et crée des résultats inattendus.

Pour finir nous aurions aimé utiliser un processus un peu moins manuel et plus robuste d'interprétation de croquis, à l'aide pas exemple de grammaires de formes et d'inférence bayésienne [10]. Ce domaine était malheureusement un peu trop éloigné du domaine de cette thèse et n'a pas pu être abordé.

5.7 Conclusion et Perspectives

Dans ce chapitre nous avons présenté un système de modélisation par croquis multi-traits capable de générer des paysages en temps réel. En se basant sur des connaissances apriori nous arrivons à analyser et générer un terrain

de manière rapide. Le terrain ainsi généré peut être édité depuis un autre point de vue sans modifier la projection par rapport au dessin original. Enfin l'édition 3D et l'édition 2D peuvent être effectuées dans un ordre arbitraire, facilitant l'utilisation de notre application comme outil de création.

Dans l'avenir nous souhaitons améliorer l'analyse des silhouettes pour l'estimation de la forme des montagnes. Améliorer le générateur de terrain pour que les contraintes de silhouettes soient mieux prises en compte est aussi un axe d'amélioration important. Enfin nous pensons l'introduction d'un système dynamique pour calculer les positions des silhouettes aiderai à simplifier la création des contraintes 3D et améliorerai la robustesse du processus.

Conclusion

Travaux présentés durant cette thèse

La première contribution de cette thèse a été un système volumique de modélisation, par croquis, qui utilise une métaphore de peinture plutôt qu'une métaphore de dessin. Les résultats du point de vue de notre cible d'utilisateurs étaient satisfaisants car ce système a permis à des débutants de créer leurs premières formes 3D.

Ce travail a motivé le développement d'une nouvelle technique de mélange de surfaces implicites : le mélange local, de manière à ce que deux surfaces implicites ne se mélangent qu'à proximité des zones ou elles s'intersectent. Cette méthode a été suivie d'une autre, le mélange à l'aide des gradients, développée à Toulouse (Voir la liste complète de mes publications).

Dans la seconde partie de cette thèse nous avons présenté des travaux sur la modélisation de surfaces, en particulier la modélisation de terrains. Nous avons présenté un modeleur de terrains temps réel, qui permet de sculpter des terrains, de manière assez naturelle, à grande échelle, à l'aide de primitives vectorielles.

A partir de ce travail nous avons proposé une méthode de génération de paysages de montagne, par croquis, vu de la première personne.

Nous avons cherché, tout au long de la thèse, à développer des interfaces et des méthodes qui puissent servir la créativité des artistes et du grand public. Car là se trouvent certains des grands défis de l'informatique graphique.

Futur de la création intuitive

De mon point de vue il y a encore au moins deux grands défis, pour la recherche en informatique graphique.

Le premier, est de rendre la création transparente pour les artistes. Comme le disait Rob Cook dans son Key-Note à Siggraph Asia en 2009 : "'le défi est de rendre la création numérique aussi transparente pour les artistes qu'on a rendu les effets spéciaux transparents pour le grand public"'. Cela s'adresse aux professionnels en particulier, pour améliorer créativité et rentabilité de la création de contenu numérique.

Mais, de mon point de vue il existe un domaine encore plus intéressant qui trouve tout son sens dans la démocratisation en puissance des imprimantes 3D. Bientôt la création numérique ne touchera plus uniquement les artistes, elle touchera le grand public. Chacun de nous sera alors tenté de personnaliser les objets qu'il utilise dans la vie de tous les jours : bibelots, accessoires, bijoux. Ces imprimantes permettront également la création de pièces sur mesure pour les réparations, comme c'est déjà le cas pour la médecine.

Les créateurs qui voudrons non seulement vendre leurs créations, qui seront soit faites sur mesure, soit elles même personnalisable par l'acheteur. Une robe, une chaussure, un bijoux, le tout sur mesure, parfaitement adapté à la morphologie de l'acheteur. Des créateurs voudrons pouvoir offrir à leurs clients des outils simples de personnalisation, ou bien le contraire.

Toutes ces applications sont actuellement hors de portée du grand public, faute d'outils. Il est déjà aisément possible de faire imprimer un objet par internet, ou même d'acheter une imprimante 3D à bas prix. Par contre il n'existe pas encore de logiciel de création 3D suffisamment simple et intuitif pour être utilisé par des novices. C'est pour ces raisons que démocratiser la création 3D est un défi passionnant qui mérite plus que jamais d'être réalisé.

Publications

[1] Adrien Bernhardt, Loic Barthe, Marie-Paule Cani, and Brian Wyvill. Implicit blending revisited. In *Computer Graphics Forum*, volume 29, pages 367–375. Wiley Online Library, 2010.

[2] Adrien Bernhardt, André Maximo, Luiz Velho, Houssam Hnaidi, and M-P Cani. Real-time terrain modeling using cpu-gpu coupled computation. In *Graphics, Patterns and Images (Sibgrapi), 2011 24th SIBGRAPI Conference on*, pages 64–71. IEEE, 2011.

[3] Adrien Bernhardt, André Maximo, Luiz Velho, Houssam Hnaidi, and Marie-Paule Cani. Real-time Terrain Modeling using CPU-GPU Coupled Computation, August 2011. ACM SIGGRAPH 2011 Poster.

[4] Adrien Bernhardt, Adeline Pihuit, Marie-Paule Cani, and Loïc Barthe. Matisse : Painting 2d regions for modeling free-form shapes. In *Proceedings of the Fifth Eurographics conference on Sketch-Based Interfaces and Modeling*, pages 57–64. Eurographics Association, 2008.

[5] Arnaud Emilien, Adrien Bernhardt, Adrien Peytavie, Marie-Paule Cani, and Eric Galin. Procedural generation of villages on arbitrary terrains. *The Visual Computer*, 28(6-8) :809–818, 2012.

[6] Olivier Gourmel, Loïc Barthe, Marie-Paule Cani, Bryan Wyvill, Adrien Bernhardt, Mathias Paulin, Herbert Grasberger, et al. A gradient-based implicit blend. *ACM Transactions on Graphics*, 2012.

Bibliographie

[7] A. Adamson and M. Alexa. Approximating and intersecting surfaces from points. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 230–239. Eurographics Association, 2003.

[8] A. Alexe, L. Barthe, M.-P. Cani, and V. Gaildrat. Shape modeling by sketching using convolution surfaces. In *Pacific Graphics*, Short paper, Macau, China, 2005.

[9] A. Alexe, V. Gaildrat, and L. Barthe. Interactive modelling from sketches using spherical implicit functions. In *AFRIGRAPH '04*, pages 25–34. ACM, 2004.

[10] C. Alvarado and R. Davis. Sketchread : a multi-domain sketch recognition engine. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 23–32. ACM, 2004.

[11] F. Anastacio, M. C. Sousa, F. Samavati, and J. A. Jorge. Modeling plant structures using concept sketches. In *Proceedings of the 4th international symposium on Non-photorealistic animation and rendering*, pages 105–113. ACM, 2006.

[12] A. Angelidis and M.-P. Cani. Adaptive implicit modeling using subdivision curves and surfaces as skeletons. In *Proceedings of the seventh ACM symposium on Solid modeling and applications*, pages 45–52. ACM, 2002.

[13] A. Angelidis, P. Jepp, and M.-P. Cani. Implicit modeling with skeleton curves : Controlled blending in contact situations. In *Shape Modeling International*, 2002. Banff, Canada.

[14] N. H. Anh, A. Sourin, and P. Aswani. Physically based Hydraulic Erosion Simulation on Graphics Processing Unit. In *Proceedings of the*

5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia, GRAPHITE '07, pages 257–264, New York, NY, USA, 2007. ACM.

[15] L. Barthe, N. A. Dodgson, M. A. Sabin, B. Wyvill, and V. Gaildrat. Two-dimensional potential fields for advanced implicit modeling operators. *Computer Graphics Forum*, 22(1) :23–33, 2003.

[16] L. Barthe, B. Wyvill, and E. de Groot. Controllable binary csg operators for "soft objects". *International Journal of Shape Modeling*, 10(2) :135–154, 2004.

[17] F. Belhadj. Terrain modeling : a constrained fractal model. In *Proceedings of the 5th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, pages 197–204. ACM, 2007.

[18] B. Benes and R. Forsbach. Layered data representation for visual simulation of terrain erosion. In *Computer Graphics, Spring Conference on, 2001.*, pages 80–86. IEEE, 2001.

[19] H. Bezerra, E. Eisemann, D. DeCarlo, and J. Thollot. Diffusion constraints for vector graphics. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, NPAR '10, pages 35–42, New York, NY, USA, 2010. ACM.

[20] J. Blinn. A generalization of algebriac surface drawing. *ACM Transaction on Graphics*, 1(3) :235–256, 1982.

[21] J. Bloomenthal. An implicit surface polygonizer. *Graphics gems IV*, pages 324–349, 1994.

[22] J. Bloomenthal and K. Shoemake. Convolution Surfaces. In *Computer Graphics (Proc. SIGGRAPH 91)*, volume 25, pages 251–256, August 1991.

[23] J. Bloomenthal and B. Wyvill. Interactive techniques for implicit modeling. *Computer Graphics (Proc. of SIGGRAPH 1990)*, 24(2) :109–116, 1990.

[24] J. Bloomenthal and B. Wyvill, editors. *Introduction to Implicit Surfaces*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[25] M. Botsch and L. Kobbelt. An intuitive framework for real-time freeform modeling. *ACM Transactions on Graphics (TOG)*, 23(3) :630–634, 2004.

[26] A. Bouthors and M. Nesme. Twinned meshes for dynamic triangulation of implicit surfaces. In *Proceedings of Graphics Interface 2007*, GI '07, pages 3–9, New York, NY, USA, 2007. ACM.

[27] J. Brosz, F. F. Samavati, and M. C. Sousa. Terrain synthesis by-example. *Advances in Computer Graphics and Computer Vision*, pages 58–77, 2007.

[28] E. Bruneton. Proland. `http://proland.inrialpes.fr`.

[29] E. Bruneton and F. Neyret. Real-time rendering and editing of vector-based terrains. In *Computer Graphics Forum*, volume 27, pages 311–320. Wiley Online Library, 2008.

[30] B. Buxton. *Sketching User Experiences : Getting the Design Right and the Right Design : Getting the Design Right and the Right Design*. Morgan Kaufmann, 2010.

[31] J. M. Cohen, J. F. Hughes, and R. C. Zeleznik. Harold : A world made of drawings. In *NPAR 2000 : First International Symposium on Non Photorealistic Animation and Rendering*, pages 83–90, June 2000.

[32] G. J. de Carpentier and R. Bidarra. Interactive gpu-based procedural heightfield brushes. In *Proceedings of the 4th International Conference on Foundations of Digital Games*, pages 55–62. ACM, 2009.

[33] T. Duff. Interval arithmetic and recursive subdivision for implicit functions and constructive solid geometry. In *Computer Graphics (Proceedings of SIGGRAPH 92)*, pages 131–138, 1992.

[34] e-on software. Vue. `http://www.e-onsoftware.com`.

[35] E. Ferley, M.-P. Cani, and J.-D. Gascuel. Practical volumetric sculpting. *The Visual Computer*, 16(8) :469–480, 2000.

[36] J. Flórez, M. Sbert, M. A. Sainz, and J. Vehí. Improving the Interval Ray Tracing of Implicit Surfaces. In *Computer Graphics International 2006*, pages 655–664, 2006.

[37] J. Gain, P. Marais, and W. Straßer. Terrain Sketching. In *Proceedings of the Symposium on Interactive 3D Graphics and Games*, I3D '09, pages 31–38, New York, NY, USA, 2009. ACM.

[38] Y. Gingold, T. Igarashi, and D. Zorin. Structured annotations for 2d-to-3d modeling. In *ACM Transactions on Graphics (TOG)*, volume 28, page 148. ACM, 2009.

[39] R. Goldman. Curvature formulas for implicit curves and surfaces. *Computer Aided Geometric Design*, 22(7) :632–658, 2005.

[40] O. Gourmel, A. Pajot, M. Paulin, L. Barthe, and P. Poulin. Fitted bvh for fast raytracing of metaballs. In *Computer Graphics Forum*, volume 29, pages 281–288. Wiley Online Library, 2010.

[41] R. W. Hall. Fast parallel thinning algorithms : parallel speed and connectivity preservation. *Commun. ACM*, 32(1) :124–131, 1989.

[42] H. Hnaidi, E. Guérin, S. Akkouche, A. Peytavie, and E. Galin. Feature based Terrain Generation using Diffusion Equation. *Computer Graphics Forum*, 29(7), September 2010.

[43] P. Hsu and C. Lee. The scale method for blending operations in functionally-based constructive geometry. *Computer Graphics Forum*, 22(2) :143–158, 2003.

[44] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy : a sketching interface for 3d freeform design. In *SIGGRAPH '99*, pages 409–416, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[45] R. Juchmes, P. Leclercq, and S. Azar. A multi-agent system for the interpretation of architectural sketches. In *Proceedings of the First Eurographics conference on Sketch-Based Interfaces and Modeling*, pages 53–61. Eurographics Association, 2004.

[46] O. A. Karpenko and J. F. Hughes. Smoothsketch : 3d free-form shapes from complex sketches. In *SIGGRAPH '06*, pages 589–598, New York, NY, USA, 2006. ACM.

[47] V. Kraevoy, A. Sheffer, and M. van de Panne. Modeling from contour drawings. In *Proceedings of the 6th Eurographics Symposium on Sketch-Based interfaces and Modeling*, pages 37–44. ACM, 2009.

[48] L. Leblanc, J. Houle, and P. Poulin. Modeling with blocks. *The Visual Computer*, 27(6-8) :555–563, 2011.

[49] D. Levin. Mesh-independent surface interpolation. *Geometric modeling for scientific visualization*, 3 :37–49, 2003.

[50] P. Ljung and A. Ynnerman. Extraction of intersection curves from isosurfaces on co-located 3d grids. In *The Annual SIGRAD Conference Special Theme–Real-Time Simulations November 20–21, 2003 Umeå University, Umeå, Sweden, Conference Proceedings*, page 23. Citeseer, 2003.

[51] M. Meyer, M. Desbrun, P. Schröder, A. H. Barr, et al. Discrete differential-geometry operators for triangulated 2-manifolds. *Visualization and mathematics*, 3(2) :52–58, 2002.

[52] K. Museth, D. E. Breen, R. T. Whitaker, and A. H. Barr. Level set surface editing operators. In T. Appolloni, editor, *SIGGRAPH*, pages 330–338. ACM, 2002.

[53] F. Musgrave, C. Kolb, and R. Mace. The synthesis and rendering of eroded fractal terrains. In *ACM SIGGRAPH Computer Graphics*, volume 23, pages 41–50. ACM, 1989.

[54] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa. Fibermesh : designing freeform surfaces with 3d curves. *ACM Trans. Graph.*, 26(3) :41, 2007.

[55] A. Nealen, O. Sorkine, M. Alexa, and D. Cohen-Or. A sketch-based interface for detail-preserving mesh editing. In *ACM SIGGRAPH 2007 courses*, page 42. ACM, 2007.

[56] A. Orzan, A. Bousseau, H. Winnemöller, P. Barla, J. Thollot, and D. Salesin. Diffusion curves : a vector representation for smooth-shaded images. In *ACM Transactions on Graphics (TOG)*, volume 27, page 92. ACM, 2008.

[57] S. Osher and R. Fedkiw. *Level set methods and dynamic implicit surfaces*, volume 153. Springer, 2002.

[58] A. Pasko, V. Adzhiev, A. Sourin, and V. Savchenko. Function representation in geometric modeling : concepts, implementation and applications. *The Visual Computer*, 11(8) :429–446, 1995.

[59] A. Peytavie, E. Galin, J. Grosjean, and S. Merillou. Arches : a framework for modeling complex terrains. In *Computer Graphics Forum*, volume 28, pages 457–467. Wiley Online Library, 2009.

[60] A. Pihuit, P. Kryt, and M.-P. Cani. Hands on virtual clay. In *Shape Modeling and Applications, 2008. SMI 2008. IEEE International Conference on*, pages 267–268. IEEE, 2008.

[61] A. Pihuit, O. Palombi, M. Cani, et al. Reconstruction implicite de surfaces 3d à partir de régions 2d dans des plans parallèles. In *Journées de l'AFIG*, 2009.

[62] A. Rivers, F. Durand, and T. Igarashi. *3D modeling with silhouettes*, volume 29. ACM, 2010.

[63] V. Savchenko, A. Pasko, O. Okunev, and T. Kunii. Function representation of solids reconstructed from scattered surface points and contours. In *Computer Graphics Forum*, volume 14, pages 181–188. Wiley Online Library, 1995.

[64] J. Schmid, M. S. Senn, M. Gross, and R. W. Sumner. Overcoat : an implicit canvas for 3d painting. *To appear in ACM TOG*, 30 :4, 2011.

[65] R. Schmidt, K. Singh, and R. Balakrishnan. Sketching and composing widgets for 3d manipulation. In *Computer Graphics Forum*, volume 27, pages 301–310. Wiley Online Library, 2008.

[66] R. Schmidt, B. Wyvill, and E. Galin. Interactive implicit modeling with hierarchical spatial caching. In *Shape Modeling and Applications, 2005 International Conference*, pages 104–113. IEEE, 2005.

[67] R. Schmidt, B. Wyvill, M. Sousa, and J. Jorge. Shapeshop : Sketch-based solid modeling with blobtrees. In *ACM SIGGRAPH 2006 Courses*, page 14. ACM, 2006.

[68] J. Sethian. *Level set methods and fast marching methods : evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, volume 3. Cambridge university press, 1999.

[69] A. Sherstyuk. Kernel functions in convolution surfaces : a comparative analysis. *The Visual Computer*, 15 :171–182, 1999.

[70] J. M. Snyder. Interval analysis for computer graphics. In *Computer Graphics (Proceedings of SIGGRAPH 92)*, pages 121–130, 1992.

[71] P. Software. Terragen. `http://planetside.co.uk`.

[72] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 175–184. ACM, 2004.

[73] C. Tai, H. Zhang, and J. Fong. Prototype modeling from sketched silhouettes based on convolution surfaces. *Computer Graphics Forum*, 23 :71–83, 2004.

[74] U. Trottenberg, C. W. Oosterlee, and A. Schuller. *Multigrid*. Access Online via Elsevier, 2000.

[75] E. Turquin, J. Wither, L. Boissieux, M.-P. Cani, and J. F. Hughes. A sketch-based interface for clothing virtual characters. *Computer Graphics and Applications, IEEE*, 27(1) :72–81, 2007.

[76] O. Šťava, B. Beneš, M. Brisbin, and J. Křivánek. Interactive Terrain Modeling using Hydraulic Erosion. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '08, pages 201–210, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association.

[77] N. Watanabe and T. Igarashi. A Sketching Interface for Terrain Modeling. In *ACM SIGGRAPH 2004 Posters*, SIGGRAPH '04, pages 73–, New York, NY, USA, 2004. ACM.

[78] J. Wither, F. Bertails, and M.-P. Cani. Realistic hair from a sketch. In *Shape Modeling and Applications, 2007. SMI'07. IEEE International Conference on*, pages 33–42. IEEE, 2007.

[79] J. Wither, F. Boudon, M.-P. Cani, and C. Godin. Structure from silhouettes : a new paradigm for fast sketch-based design of trees. In *Computer Graphics Forum*, volume 28, pages 541–550. Wiley Online Library, 2009.

[80] B. Wyvill, A. Guy, and E. Galin. Extending the csg tree. warping, blending and boolean operations in an implicit surface modeling system. In *Computer Graphics Forum*, volume 18, pages 149–158. Wiley Online Library, 1999.

[81] B. Wyvill and K. van Overveld. Polygonization of Implicit Surfaces with Constructive Solid Geometry. *Journal of Shape Modelling*, 2(4) :257–274, 1996.

[82] B. Wyvill and G. Wyvill. Better blending of implicit objects at different scales. In *ACM Siggraph 2000 Sketch Proceedings*. ACM, 2000.

[83] G. Wyvill, C. McPheeters, and B. Wyvill. Data Structure for Soft Objects. *The Visual Computer*, (4) :227–234, February 1986.

[84] C. Yang, D. Sharon, and M. van de Panne. Sketch-based modeling of parameterized objects. In *EG Workshop on Sketch-Based Interfaces and Modeling*, pages 63–72, 2005.

[85] J. Yu and G. Turk. Reconstructing surfaces of particle-based fluids using anisotropic kernels. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 217–225. Eurographics Association, 2010.

[86] H. Zhou, J. Sun, G. Turk, and J. M. Rehg. Terrain Synthesis from Digital Elevation Models. *IEEE Transactions on Visualization and Computer Graphics*, 13 :834–848, 2007.

[87] Q. Zhou, T. Weinkauf, and O. Sorkine. Feature-based mesh editing. In *Eurographics 2011-Short Papers*, pages 1–4. The Eurographics Association, 2011.